UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

INGENIERÍA INDUSTRIAL

Diseño de un Protocolo de Calibración de Cámaras Estéreo

Proyecto Fin de Carrera

Departamento de Ingeniería de Sistemas y Automática

Autor: José Javier Alcalde Sanz

Tutor: Arturo de la Escalera Hueso

Abril de 2014

A mis padres y hermanos

Resumen

El presente Proyecto Final de Carrera tiene como objetivo principal la elaboración completa de un protocolo que indique una manera eficaz de calibrar un sistema estéreo.

Un sistema estéreo es aquél formado por, al menos, dos cámaras; y está caracterizado por disponer de información tridimensional del entorno. Es decir, un sistema estéreo es capaz de indicar la distancia a la que se encuentran los objetos respecto a las cámaras que los han ilustrado.

En el caso actual, se va a proceder, en primer lugar, con la calibración, por separado, de cada una de las dos cámaras utilizadas (cámara Bumblebee xb3) para obtener sus parámetros más representativos, parámetros intrínsecos y extrínsecos.

Posteriormente, se tratarán las imágenes proporcionadas por las cámaras hasta lograr que se encuentren en las condiciones adecuadas para un análisis estéreo, o lo que es lo mismo, bien rectificadas, sin distorsiones, ni inclinaciones.

Por último, se comprobará la eficiencia del protocolo representando el mapa de disparidad de las imágenes.

Todo lo anterior se va a desarrollar en un entorno de programación en lenguaje C/C++. Para poder realizarlo se van a usar, como elemento de apoyo, las librerías OpenCV de visión por Computador.

Abstract:

This Thesis Project aims at fully developing a protocol which would specify an efficient way of calibrating a stereo system.

A stereo system is a system formed by at least two cameras and it is characterized by providing three dimensional information of the surrounding. In other words, a stereo system is able of indicating the distance to an object with respect to the corresponding cameras.

During the project, the first step is the calibration of each camera separately of the two used (Bumblebee xb3 cameras); in order to retrieve the camera's most representative parameters, both intrinsic and extrinsic.

Secondly, the images provided by the cameras will be edited until they are both in equal conditions for a stereo analysis, correctly rectified, without distortions or inclinations.

Finally, the efficiency of the protocol will be presented with a map of image disparities.

All the described steps are developed using the C/C++ programming environment. To be able to do this, the OpenCV libraries are used as an assistance element of computer vision.

Agradecimientos

Culminar una larga travesía ofrece la posibilidad de pararse a pensar acerca del camino recorrido, y, con ello, en los hermanos de viaje. Algunos compartieron tal solo unos pasos del trayecto, y otros aún seguirán andando conmigo cuando se termine la etapa, pero todos me han ayudado, en mayor o menor medida, a finalizar esta ruta. Por eso, me gustaría deciros algo: gracias.

A mis padres y hermanos, por los consejos aportados y el respaldo proporcionado. Porque siempre habéis estado en el lugar preciso y con las palabras apropiadas. Y, en general, a toda mi familia, por el interés mostrado por mi situación a lo largo del camino, a pesar de la distancia.

A mis compañeros de la Universidad, por los ánimos que acertadamente me habéis dado; y por el apoyo en todo tipo de circunstancias, imprescindible para cerrar con éxito el itinerario.

A mis amigos, por ser como sois; por los momentos de distensión vividos, necesarios para afrontar la rutina con energía; y por la fuerza concedida durante los tropiezos e imprevistos de la marcha.

A mi tutor, por brindarme la oportunidad de realizar este proyecto y aconsejarme para llevarlo a cabo de la mejor manera posible. Asimismo, no me quiero olvidar de las personas del laboratorio LSI que me han ayudado altruistamente a captar las imágenes que he requerido.

A todos, gracias.

Javier

Índice

Índice de Tablas

Índice de Figuras

1. Introducción

Una de las primeras aplicaciones en la mejora de la calidad de las imágenes fué el tratamiento de las mismas para su publicación en periódicos, enviadas por cable submarino entre Londres y Nueva York. A principios de los años veinte, el tiempo de transmisión de las propias imágenes se vió reducido considerablemente al pasar de más de una semana a menos de tres horas. Ése fue uno de los primeros avances que se implementaron en el tratamiento de imágenes, pero ni mucho menos el único.

Durante las décadas siguientes, se trabajó duramente para obtener mejores resultados en un nuevo campo llamado visión artificial, basado en el tratamiento de imágenes, con la finalidad de simular el sentido artificial de la vista. Pero sería a partir de los años ochenta cuando, gracias al desarrollo vertiginoso de los ordenadores, se empezarían a dar pasos agigantados en cuanto a soluciones se refiere. Dicho desarrollo fué debido principalmente al aumento de la capacidad de cálculo de las computadoras y la disminución de su precio (haciéndolo accesible a la mayor parte de la población), así como a la aparición de hardware específico para el procesamiento de imágenes. Con ello empezaron a ser utilizables aplicaciones ya resueltas con anterioridad, pero con un tiempo de cálculo inviable ó un precio prohibitivo.

La visión artificial es una disciplina de creciente y continuo interés en el campo científico-técnico como consecuencia de la importancia y número de las posibles aplicaciones, entre las que se encuentran: robótica, procesos de inspección automática (control de calidad), navegación autónoma de vehículos, control de tráfico, análisis de imágenes médicas, vigilancia de edificios, aplicaciones militares, etc..

Para entenderlo mejor, la visión artificial, también denominada visión por computador, se basa en el análisis de imágenes a través de un ordenador para obtener información del mundo físico captado por la cámara.

En la visión por computador, la escena tridimensional es vista por una, dos o más cámaras para producir imágenes monocromáticas ó en color. Las imágenes adquiridas pueden ser segmentadas para obtener de ellas características de interés tales como bordes ó regiones. Posteriormente, de las características se obtienen las propiedades subyacentes mediante el correspondiente proceso de descripción, tras lo cual se consigue la estructura de la escena tridimensional requerida por la aplicación con la que se esté trabajando.

En cualquier caso, para llevar a cabo dicho análisis es imprescindible la adquisición de las imágenes digitales, que procedan de un archivo en el que estén precargadas, ó que sean tomadas directamente desde una cámara.

Previamente al uso de las imágenes por el programa, se tendrá que eliminar la distorsión de las mismas, mediante, por ejemplo, la aplicación de los parámetros intrínsecos de la cámara. Como se verá más adelante, ésto se puede realizar de manera sencilla con la ayuda de las librerías *opencv*.

Una vez se disponga de las imágenes, se deberán tratar con el objeto de utilizarlas en las mejores y más favorables condiciones de trabajo para la aplicación a desarrollar. Dicho tratamiento consiste principalmente, y como se irá desarrollando a lo largo del presente Proyecto Fin de Carrera, en:

- Obtener los parámetros intrínsecos de las cámaras con las que se han tomado las imágenes.
- Eliminar la distorsión de las imágenes.
- Obtener los puntos de interés de las imágenes.
- Obtener la matriz fundamental y las homografías.
- Rectificar las imágenes.
- Calcular la disparidad de los objetos que aparecen en la imagen estéreo.

Para llevar a cabo el proyecto se ha precisado de las características técnicas de las cámaras que se han utilizado, una *BumbleBee XB3* prestada por la Universidad Carlos III de Madrid con la que se ha efectuado la correcta realización del estudio. También, han servido de apoyo la herramienta *visual studio* para el desarrollo del código en lenguaje C; y las librerías *opencv* de libre adquisición.

1.1. Objetivos:

El objetivo principal del presente Proyecto Fin de Carrera es el Diseño de un Protocolo de Calibración de Cámaras Estéreo con la finalidad de obtener unas imágenes adecuadas con las que poder trabajar para el tratamiento y desarrollo de aplicaciones en el campo de la visión estéreo. Para poder materializarlo eficazmente se seguirá la secuencia previamente descrita y que a continuación se repite:

- Obtener los parámetros intrínsecos de las cámaras con las que se han tomado las imágenes.
- Eliminar la distorsión de las imágenes.
- Obtener los puntos de interés de las imágenes.
- Obtener la matriz fundamental y las homografías.
- Rectificar las imágenes.
- Calcular la disparidad de los objetos que aparecen en la imagen estéreo.

1.2. Estructura:

Para facilitar la comprensión del proyecto, se ha estructurado y redactado de forma clara y ordenada.

En primer lugar, se ha elaborado un desarrollo teórico de todos los conceptos de óptica y visión por computador que se han estimado necesarios para la correcta compresión de los puntos a tratar.

En segundo lugar, están explicados los pasos que he llevado a cabo para completar este proyecto, prestando especial atención a los resultados que he obtenido durante los meses de trabajo.

Seguidamente, se presenta el apartado con las conclusiones y con las posibles líneas de trabajo futuro.

En el cuarto capítulo, está expuesto el presupuesto estimado para la ejecución del proyecto.

Para terminar se muestran los apartados correspondientes a la bibliografía, y los anexos.

Durante la lectura de todo el proyecto se pueden apreciar fragmentos del código, implementado en lenguaje C, que he desarrollado para la obtención de los resultados.

2. Marco teórico

2.1. Calibración:

La calibración de la cámara tiene como finalidad obtener todos los parámetros que son necesarios para la formación de la imagen, proceso esquematizado en la figura 2.1. Por lo tanto, mediante los algoritmos de calibración obtendremos tanto los parámetros geométricos, como los radiométricos. De los segundos, los radiométricos, únicamente mencionar que son los relacionados con el brillo de los objetos proyectados, como, por ejemplo, la apertura de la óptica, o el tiempo de exposición y la ganancia del amplificador de la cámara. Durante el presente proyecto se hará hincapié en los primeros, dado que para problemas de visión por computador son los de mayor interés.

A grandes rasgos los parámetros geométricos se pueden clasificar en dos tipos: parámetros intrínsecos y parámetros extrínsecos. Aunque previamente a calcular cada uno de ellos se debe explicar el funcionamiento básico de las cámaras.

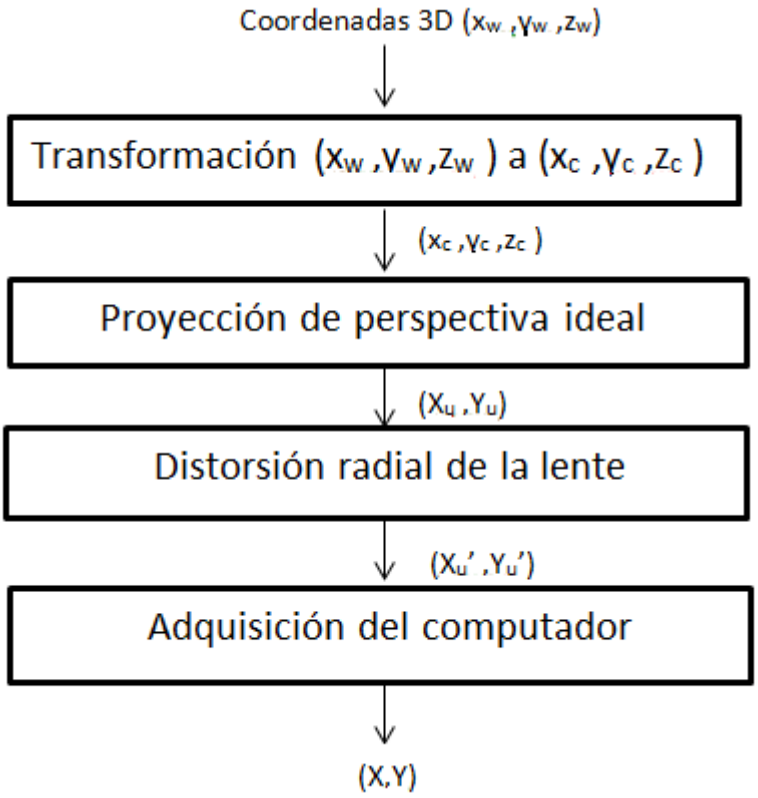


Figura 2.1. *Proceso de adquisición de una imagen*

Donde: (x_w ,y_w ,z_w) son las coordenadas 3D del punto del objeto en el sistema de coordenadas del mundo. (x_c ,y_c ,z_c) son las coordenadas 3D del punto objeto en el sistema de coordenadas de la cámara. (X_u ,Y_u) son las coordenadas del punto imagen teniendo en cuenta el modelo de cámara con apertura infinitesimal. (X_u',Y_u') son las coordenadas del punto imagen una vez corregida la distorsión de la lente. (X,Y) son las coordenadas utilizadas en el ordenador.

2.1.1. Modelo Cámara:

El dispositivo más simple de formación de imágenes es una cámara con una apertura infinitesimalmente pequeña. Esta cámara, para entenderlo de forma simple, se puede asemejar a un cubo oscuro con un agujero infinitesimal en uno de sus lados. Como se puede observar en la figura 2.2, los rayos de luz entran en el cubo atravesando dicha apertura y forman una imagen en la cara interior del cubo opuesta al lado del orificio. Al tratarse de un agujero tan pequeño, de los rayos de luz que proceden de cada punto del mundo real (3D) exclusivamente uno pasa por él, lo que implica que sobre cada punto de la cara interior del cubo situada enfrente de la apertura solo incide un rayo de luz, formando así una imagen en el plano (2D). De esta manera todos los puntos del espacio estarán perfectamente enfocados. De igual modo, la imagen que se forme se verá invertida y a una longitud denominada distancia focal, *f*. Esta forma de obtener la imagen es comúnmente conocida como modelo *pin-hole*.

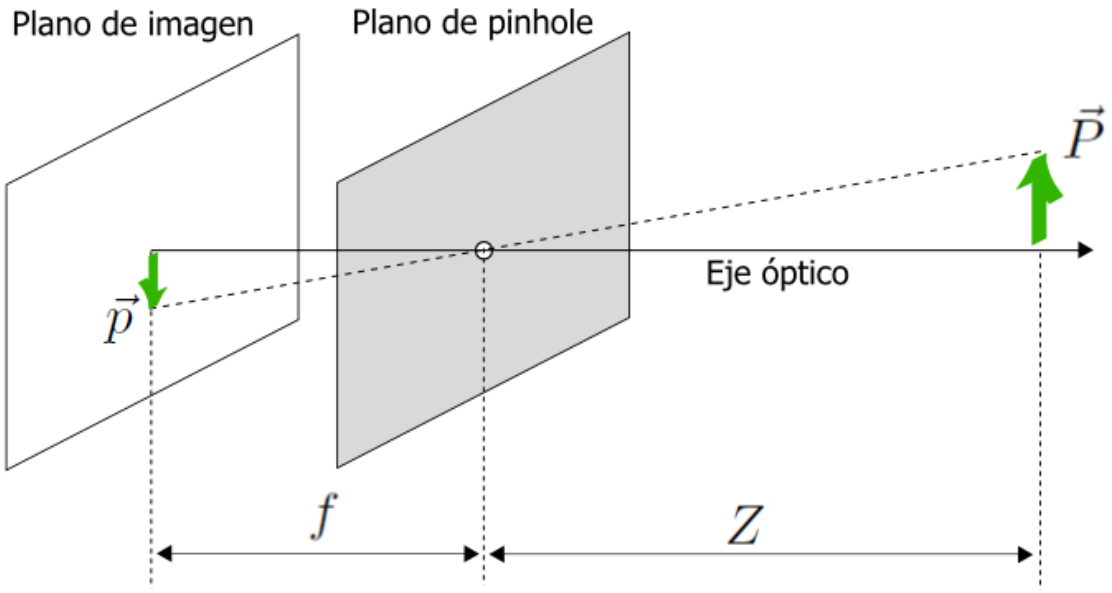


Figura 2.2. *Modelo pin-hole*

Si nos fijamos en la representación geométrica equivalente del modelo *pin-hole*, expuesta en la figura 2.3, se pueden deducir las ecuaciones que propone dicho modelo para relacionar un punto situado en el espacio (3D) con su proyección en la imagen (2D). Estas ecuaciones se logran fácilmente mediante una relación de semejanza de triángulos, quedando:

$$x = \frac{f}{Z}X \qquad (2.1)$$

$$y = \frac{f}{Z}Y \qquad (2.2)$$

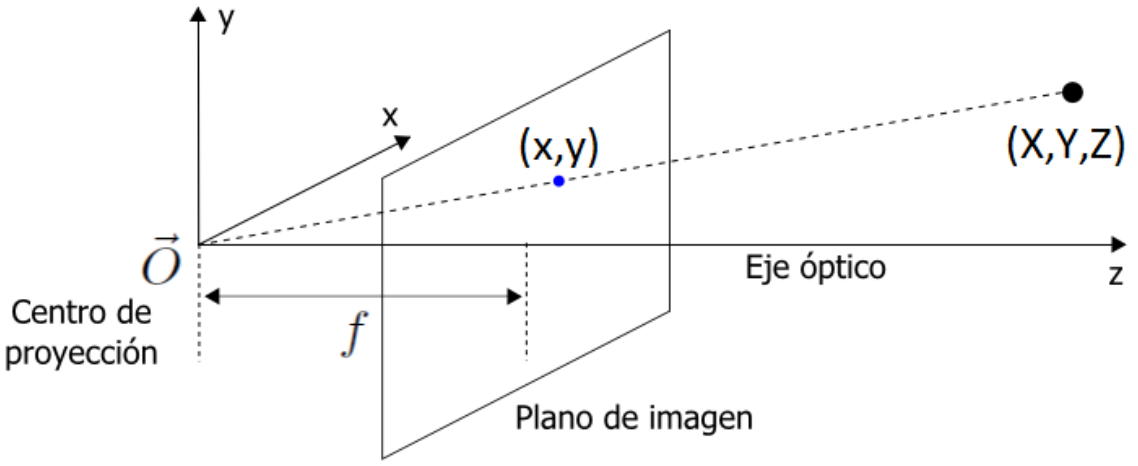


Figura 2.3. *Geometría del modelo pin-hole*

Al resultar tan sencillo, es un modelo que se utiliza en numerosas aplicaciones de visión por computador. Aunque en realidad, el único parámetro que modela es la distancia focal, que se obtiene despejando de cualquiera de las dos ecuaciones anteriores:

$$f = \frac{x}{X}Z = MZ \qquad (2.3)$$

Donde el nuevo parámetro *M* es el factor de Magnificación.

Para poder obtener más características habría que hacer más grande la apertura, pero ello implica que la imagen se desenfoca. Para evitar este problema se pueden diseñar otros sistemas, como, por ejemplo, el modelo de lente fina (figura 2.4) que incluye una lente mediante la que se canalizan todos los rayos luminosos procedentes de un determinado punto del espacio.

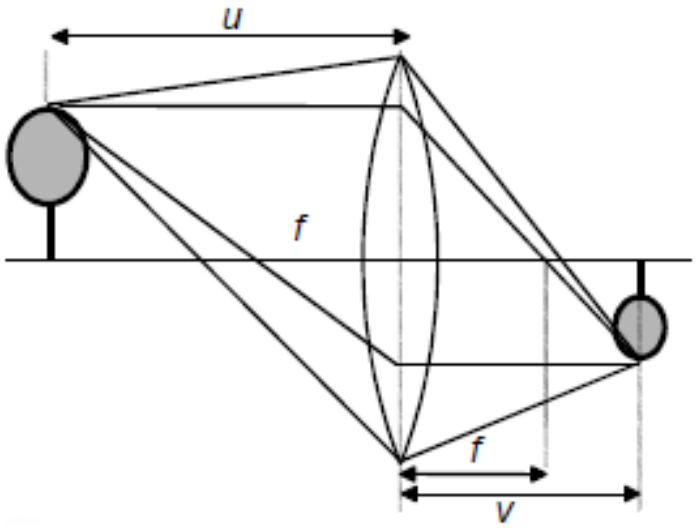


Figura 2.4. *Modelo lente fina*

En general, puede afirmarse que la relación existente entre cualquier punto *p* del espacio y su proyección *p'* en el plano de la imagen se puede simplificar con la siguiente ecuación:

$$\begin{pmatrix} p' \\ 1 \end{pmatrix} = K \begin{pmatrix} R & t \\ 0 \quad 0 \quad 0 & 1 \end{pmatrix} \begin{pmatrix} p \\ 1 \end{pmatrix} \qquad (2.4)$$

Donde *K* es una matriz 3x3 en la que se representan los parámetros intrínsecos de la cámara. *R* es una matriz de rotación 3x3 y *t* es un vector de traslación de la cámara con sistema de referencia en el mundo. *R* y *t* son los parámetros extrínsecos de la cámara.

2.1.2. Parámetros Intrínsecos:

Los parámetros intrínsecos son todos los que tienen relación con el conjunto de cámara y óptica, es decir, son los que están involucrados en la transformación de puntos 3D en el sistema de referencia de la cámara a puntos 2D del plano imagen. Dentro de este tipo se incluyen el desplazamiento del centro de la imagen, la distancia focal, los coeficientes de distorsión y el tamaño de cada píxel.

Una forma común de calcularlos es, adoptando en primera instancia el modelo *pin-hole*, descrito anteriormente, para en pasos posteriores ir aplicando transformaciones que permitan obtener todos los parámetros.

Los parámetros intrínsecos se pueden agrupar mayoritariamente dentro de una matriz, denominada matriz de calibración para facilitar su posterior tratamiento matemático. Dicha matriz tiene la forma:

$$K = \begin{pmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \qquad (2.5)$$

Cuyos elementos se exponen a continuación:

f_x y f_y son las longitudes focales de la lente de la cámara en el eje x y en el eje y respectivamente. Estos parámetros se miden en píxeles. Y se obtienen según las ecuaciones:

$$f_x = f S_x \qquad (2.6)$$

$$f_y = f S_y \qquad (2.7)$$

Donde f es la distancia focal de la lente de la cámara medida en unidades de longitud. Y S_x y S_y son el número de píxeles por unidad de longitud del sensor en el eje x y en el eje y respectivamente.

s es un factor que indica el grado de perpendicularidad de las paredes de los píxeles del sensor. Tiene un valor inversamente proporcional a la tangente del ángulo que forma el eje x con el eje y, lo que implica que generalmente se puede asumir como nula su aportación en la matriz, debido a que en la mayor parte de los sensores actuales los píxeles son rectangulares.

Para terminar, c_x y c_y determinan el desplazamiento del centro de la imagen.

El hecho de incorporar una lente a la cámara para evitar la problemática comentada anteriormente con el modelo *pin-hole* da pie a presentar el último parámetro intrínseco a tener en cuenta: la distorsión de la lente. Debido a ella, se deteriora la calidad geométrica de la imagen hasta el grado de que se pierde parte de la capacidad de medir las posiciones de los objetos con exactitud, aunque se mantienen todos los puntos de la propia imagen enfocados.

Dicha distorsión surge porque los rayos de luz no emergen de las lentes en las direcciones previstas por las leyes ópticas. Se pueden distinguir dos tipos: radial y tangencial.

La distorsión radial de la lente provoca que los puntos de las imágenes se desplacen de forma radial a partir del eje óptico, como se puede observar en la figura 2.5. Es decir, favorece que los rayos más alejados del centro óptico se curven bastante más que aquellos que inciden directamente en las proximidades del centro de la lente. Su causa más común es un pulido defectuoso de la propia lente.

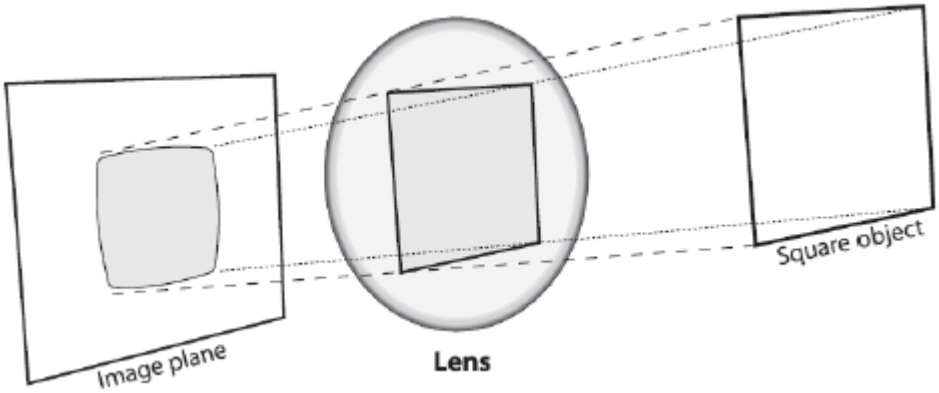


Figura 2.5. *Distorsión radial*

Para modelar matemáticamente con la máxima precisión la distorsión radial se requeriría una sucesión infinita de elementos. Por el contrario, en la práctica, y para la inmensa mayoría de las aplicaciones dentro de la visión artificial, con tener en cuenta los dos primeros términos de la serie, o incluso únicamente el primero, es suficiente. De esta manera, la ecuación que relaciona los píxeles con distorsión y los que carecen de ella, quedaría:

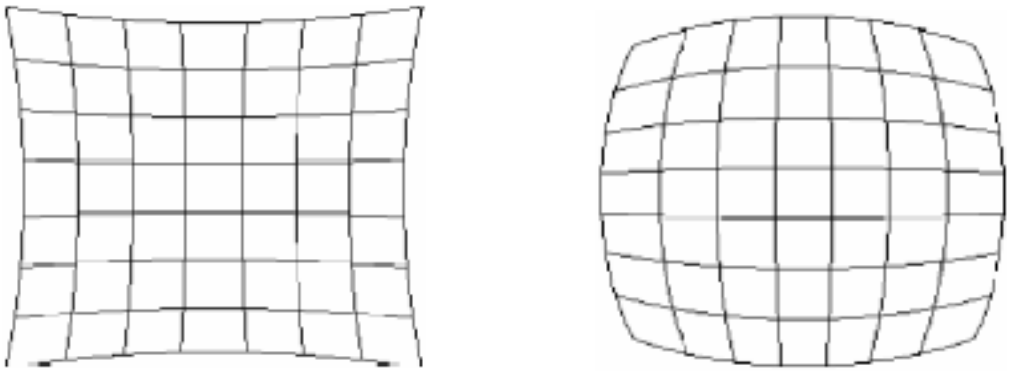
$$x_u = x_d(1 + k_1 r^2 + k_2 r^4) \qquad (2.8)$$

$$y_u = y_d(1 + k_1 r^2 + k_2 r^4) \qquad (2.9)$$

$$r = \sqrt{(x_d^2 + y_d^2)} \qquad (2.10)$$

Donde k_1 y k_2 son los coeficientes de distorsión; (x_d, y_d) son las coordenadas del píxel distorsionado; y (x_u, y_u) son las coordenadas del píxel corregido.

El grado de distorsión de la imagen se determina por el valor de los coeficientes de distorsión. Según sus signos se puede hablar de dos tipos de distorsión radial: positiva y negativa. La distorsión positiva se refleja en la formación de una imagen de tipo cojín, mientras que la negativa produce una imagen de tipo barril. En la figura 2.6 se pueden apreciar las diferencias entre los dos tipos de distorsiones radiales.



Distorsión tipo cojín Distorsión tipo barril

Figura 2.6. *Tipos de geometrías de distorsión radial*

Por otro lado, la distorsión tangencial se produce perpendicularmente a las líneas radiales a partir del eje óptico. Su principal causa es un centrado defectuoso de todos los elementos que configuran el sistema de lentes, debido a un montaje imperfecto de los elementos que la componen, como se puede apreciar, a continuación, en la figura 2.7.

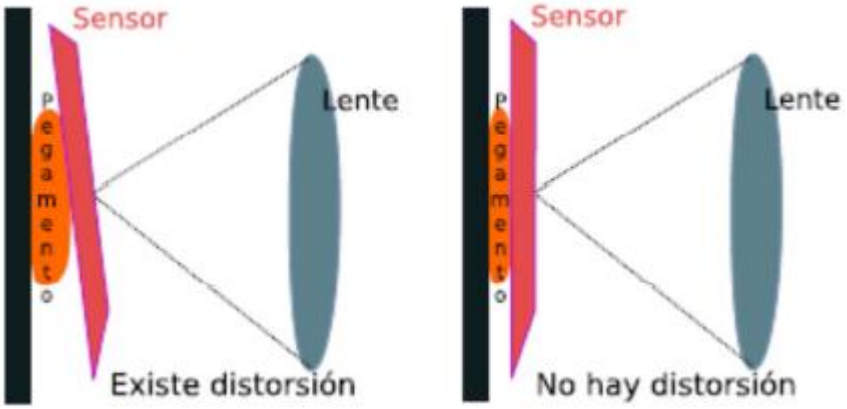


Figura 2.7. *Distorsión tangencial*

Los efectos de la distorsión tangencial son menos significativos que los de la distorsión radial. Por ello, en la mayor parte de las aplicaciones, se considerarán despreciables, teniendo únicamente en cuenta los radiales.

Al igual que la radial, la forma más precisa de modelar la distorsión tangencial también es como una sucesión infinita de elementos. Aunque, de la misma manera que en la anterior, se puede aproximar matemáticamente, sin riesgo a cometer un grave error, por una serie con dos términos. Así, las ecuaciones que relacionan los píxeles quedarían:

$$x_u = x_d + (2p_1 y_d + p_2(r^2 + 2x_d^2)) \qquad (2.11)$$

$$y_u = y_d + (2p_2 x_d + p_1(r^2 + 2y_d^2)) \qquad (2.12)$$

Donde p_1 y p_2 son los coeficientes de distorsión; y al igual que antes (x_d, y_d) son las coordenadas del píxel distorsionado; (x_u, y_u) son las coordenadas del píxel corregido; y r viene definido en la ecuación (2.10).

Todos los coeficientes de distorsión, radiales y tangenciales, se pueden agrupar en un único vector, denominado vector de distorsión, para trabajar matemáticamente de una manera más sencilla.

En resumen, los parámetros intrínsecos de la cámara los conforman tanto los pertenecientes a la matriz de calibración, como al vector de distorsión, quedando completamente definidos si se conocen los términos de ambos.

2.1.3. Parámetros Extrínsecos:

Los parámetros extrínsecos son los que hacen referencia a la colocación física de la cámara, es decir, los que definen la posición y la orientación de la cámara con respecto al sistema de coordenadas global, pues no dependen de la propia cámara sino de su disposición espacial. Dentro de éstos se incluyen todos los componentes del vector de traslación y los de las matrices de rotación de la cámara con respecto a cada uno de los ejes, como ya se mencionó anteriormente al final del apartado 2.1.1.

Como se apreció en la ecuación 2.4, que muestra la relación que existe entre cualquier punto del espacio con la representación de ese mismo punto en el plano de la imagen, tanto en función de los parámetros intrínsecos como de los extrínsecos, éstos últimos vienen determinados por el vector *t* y por la matriz *R*.

En la figura 2.8 se puede observar la rotación en sentido horario de un punto cualquiera alrededor de cada uno de los ejes coordenados, debido a que, dependiendo del eje que se adopte como referencia, para realizar la mencionada rotación habrá que utilizar una matriz de rotación u otra, como se verá a continuación.

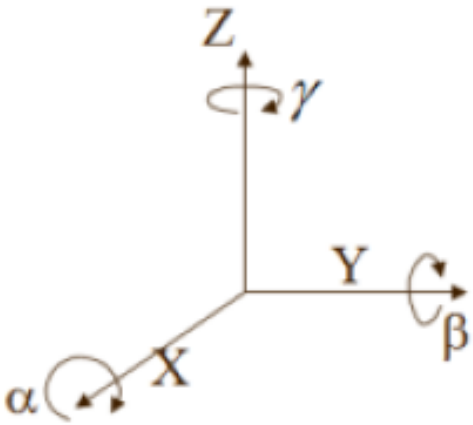


Figura 2.8. *Rotación de un punto alrededor de cada uno de los ejes*

La matriz de rotación *R* reproduce un giro de la cámara o de un objeto respecto de ella. Y, como se acaba de mencionar, adoptará distintas formas según el eje coordenado de referencia:

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha \\ 0 & -\sin\alpha & \cos\alpha \end{bmatrix} \qquad (2.13)$$

$$R_y = \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix} \qquad (2.14)$$

$$R_z = \begin{bmatrix} \cos\gamma & \sin\gamma & 0 \\ -\sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad (2.15)$$

En caso de que el giro se realizase con respecto a cualquiera de los tres ejes, se procedería de una manera similar para obtener las nuevas coordenadas, únicamente habría que cambiar la matriz de rotación empleada. Por ejemplo, si el giro tuviera lugar respecto al eje Z, como se muestra en la figura 2.9, la matriz utilizada sería R_z *y* las nuevas coordenadas quedarían:

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{bmatrix} \cos\gamma & \sin\gamma & 0 \\ -\sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} \qquad (2.16)$$

$$\begin{cases} x' = x\cos\gamma + y\sin\gamma \\ y' = y\cos\gamma - x\sin\gamma \\ z' = z \end{cases} \qquad (2.17)$$

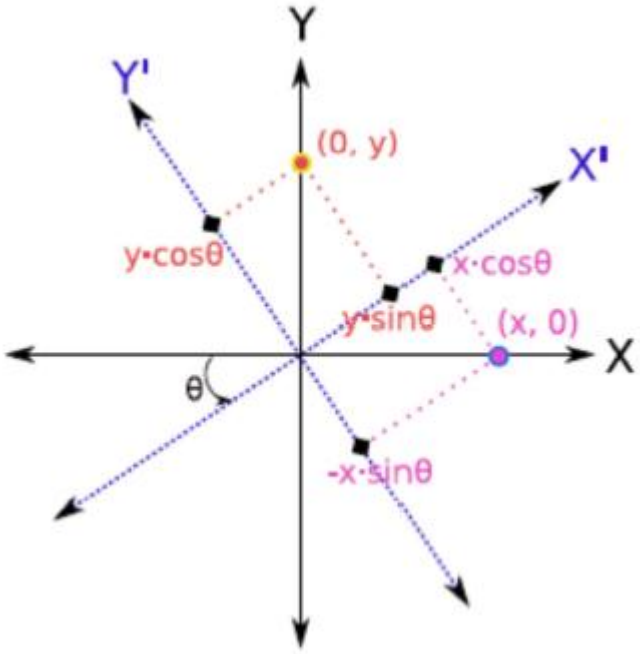


Figura 2.9. *Ejemplo de rotación respecto al eje Z*

Por otro lado, el vector t se define como un vector de traslación que reproduce un cambio en la posición del sistema de coordenadas de la cámara o de un objeto con respecto a ella.

$$t = \begin{pmatrix} t_x \\ t_y \\ t_z \end{pmatrix} \qquad (2.18)$$

Donde t_x, t_y y t_z son los desplazamientos que tienen lugar en cada uno de los ejes coordenados.

2.2. Visión estéreo:

Disponiendo de una única cámara se puede afirmar con exactitud la correspondencia entre un punto del espacio real tridimensional y el píxel en el que se ve reflejado el mismo punto en el plano de la imagen en 2D. Es decir, se pueden conocer los píxeles de la imagen que representan las coordenadas de cada punto concreto del espacio.

Sin embargo, si se quisiera realizar el proceso inverso resultaría imposible determinar, sin riesgo de error, la correspondencia existente entre cada píxel concreto de la imagen y el punto del espacio tridimensional al que representa.

Por consiguiente, si se quiere obtener dicha equivalencia partiendo de la imagen en 2D, es imprescindible la utilización de, al menos, dos cámaras. Las dos cámaras deben permitir capturar dos imágenes (una cada cámara) del mismo entorno en el mismo instante, lo que se traduce en un ligero desplazamiento de las imágenes.

La visión estéreo constituye un procedimiento para la obtención de la información tridimensional de una escena a través del tratamiento de imágenes. La forma de los objetos se determina mediante la distancia de los propios objetos en relación a un sistema de referencia.

El método para hallar la distancia a los objetos basado en la visión estereoscópica, que a continuación se especifica, comienza con un proceso de triangulación.

2.2.1. Triangulación:

El proceso de triangulación está fundamentado en las diferentes proyecciones de los objetos sobre los planos de la imagen.

Para comprenderlo de una manera más sencilla, se puede asimilar que las dos cámaras están perfectamente colocadas entre sí en dos líneas paralelas. Por otro lado, ambas cámaras comparten el mismo plano de imagen, como queda ilustrado con más detalle en la figura 2.10.

Independientemente de la otra, cada cámara tiene sus propios parámetros, tanto intrínsecos como extrínsecos, que habrá que conocer o calcular.

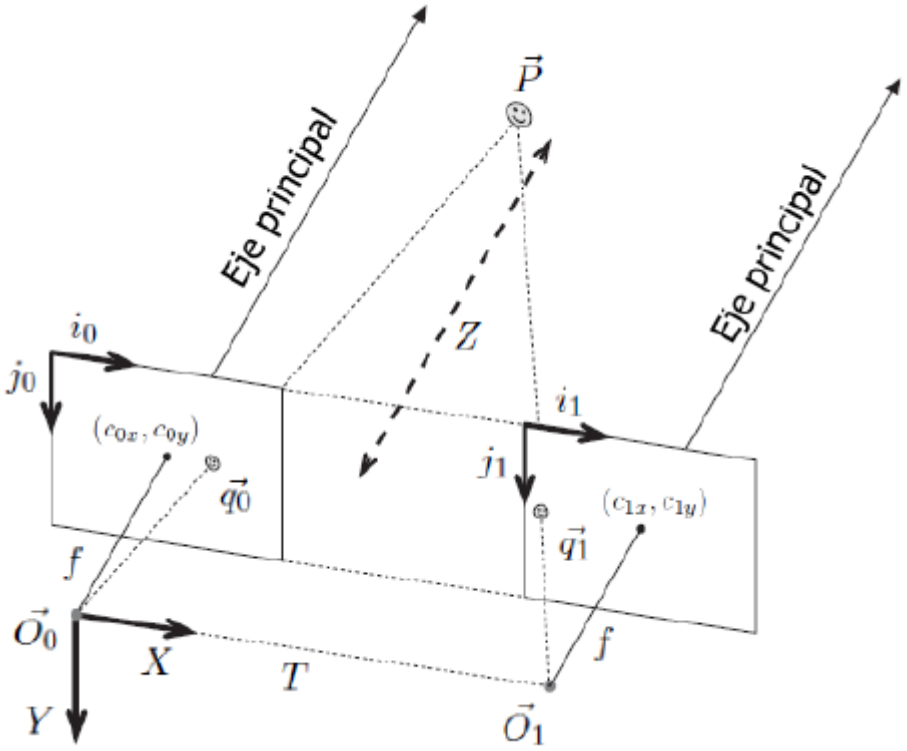


Figura 2.10. *Triangulación de cámaras en paralelo*

Los subíndices 0 hacen referencia a las definiciones relativas a la cámara situada en el lado izquierdo y los subíndices 1 se identifican con las de la cámara de la derecha.

Como se puede apreciar en la figura 2.10, las dos cámaras captan el mismo punto del espacio. Dicho punto se proyecta como q_0 y q_1 según lo haga en la imagen izquierda o en la derecha.

$$\overrightarrow{q_0} = (i_0, j_0) \qquad (2.19)$$

$$\overrightarrow{q_1} = (i_1, j_1) \qquad (2.20)$$

Si se dispone de las distancias focales de ambas cámaras, en la figura 2.10 por simplicidad se suponen iguales de valor constante f, se pueden calcular aplicando triangulación:

$$\frac{f}{Z} = \frac{(i_0 - c_{0x})}{X} = \frac{(i_1 - c_{1x})}{X - T} = \frac{(c_{0y} - j_0)}{Y} = \frac{(c_{1y} - j_1)}{Y} \qquad (2.21)$$

Las coordenadas en el espacio del punto *P*:

$$X = \frac{(i_0 - c_{0x})T}{d-(c_{0x}-c_{1x})} \qquad (2.22)$$

$$Y = \frac{(c_{0y} - j_0)T}{d-(c_{0x}-c_{1x})} \qquad (2.23)$$

$$Z = \frac{f \cdot T}{d-(c_{0x}-c_{1x})} \qquad (2.24)$$

$$d = i_0 - i_1 \qquad (2.25)$$

Donde (c_{ox}, c_{oy}) y (c_{1x}, c_{1y}) se identifican con las coordenadas del centro óptico de las cámaras izquierda y derecha respectivamente; *d* representa la disparidad de la imagen; y *T* hace referencia a la distancia entre las dos cámaras, la cual permanecerá constante.

En la configuración específica de la figura 2.10 para conocer la posición en el espacio tridimensional de un punto cualquiera *P* será necesario disponer de los parámetros intrínsecos y extrínsecos de las cámaras e identificar el mismo punto proyectado en ambas imágenes. Ésto es debido a que un mismo punto del espacio se proyectará a la misma altura en las dos imágenes:

$$j_0 - c_{oy} = j_1 - c_{1y} \qquad (2.26)$$

En el caso en el que las cámaras tengan el centro óptico en la misma posición las ecuaciones que determinan las coordenadas en el espacio del punto *P* se simplifican:

$$X = \frac{(i_0 - c_{0x})T}{d} \qquad (2.27)$$

$$Y = \frac{(c_{0y} - j_0)T}{d} \qquad (2.28)$$

$$Z = \frac{f \cdot T}{d} \qquad (2.29)$$

De las ecuaciones anteriores, cabe hacer hincapié en la relación de proporcionalidad inversa que se deduce entre la profundidad de un objeto y su disparidad en la imagen. De esta manera, los objetos más próximos a las cámaras producen unas disparidades mayores a las de los que están situados en distancias más alejadas.

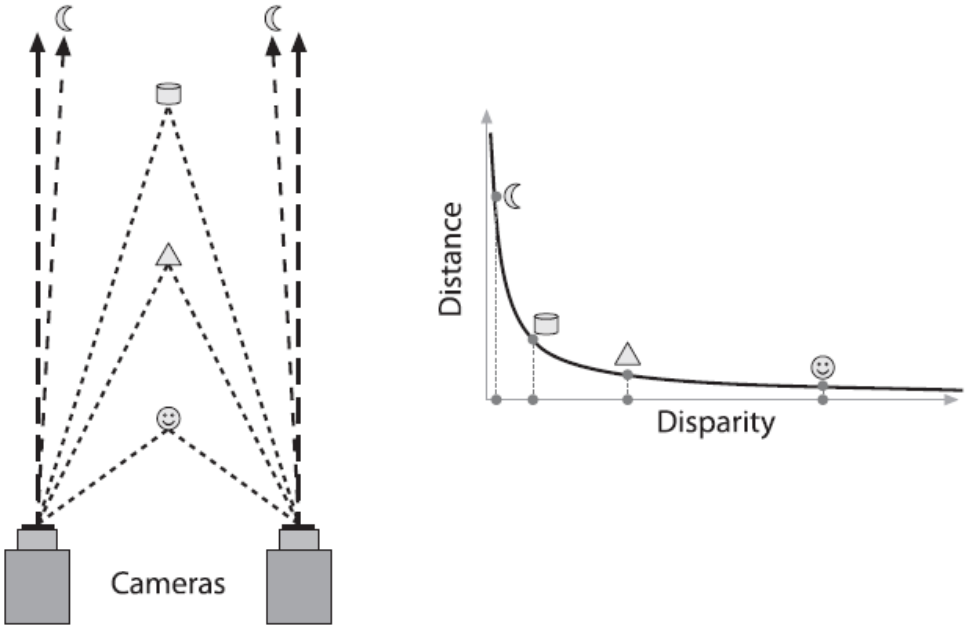


Figura 2.11. *Relación entre distancia y disparidad*

2.2.2. Geometría epipolar:

Es muy difícil, en la realidad, conseguir una configuración ideal como la expuesta anteriormente en la que los ejes de las dos cámaras estén perfectamente alineados en paralelo y la línea base sea perpendicular a los dos ejes ópticos. Por ello, es necesario recurrir a la geometría epipolar para lograr obtener la profundidad de los objetos en la imagen.

La geometría epipolar permite relacionar las proyecciones en la imagen derecha con las de la imagen izquierda del conjunto de puntos de todo el espacio capturado por las dos cámaras, habiendo sufrido una cámara con respecto a la otra, tanto una rotación como una traslación.

En la figura 2.12 se puede apreciar el principio fundamental en el que está basada la geometría epipolar.

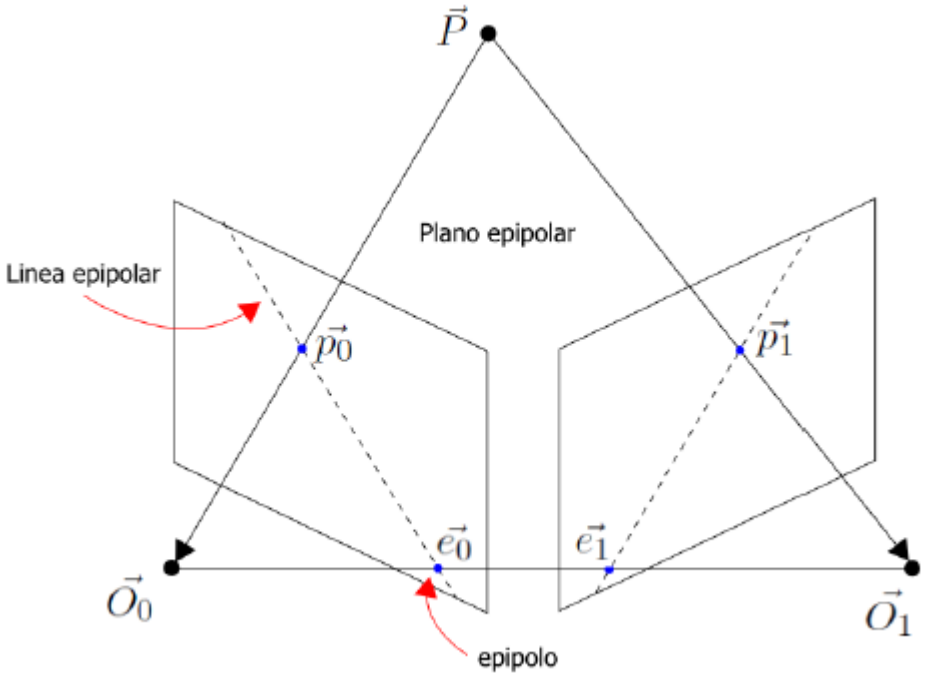


Figura 2.12. *Geometría epipolar*

Así, *P* es un punto cualquiera del espacio. Los puntos p_0 y p_1 son las proyecciones generadas por el punto *P* del mundo físico sobre los planos de cada una de las cámaras, izquierda y derecha, respectivamente. O_0 y O_1 se denominan centros ópticos o de proyección de las cámaras. Los puntos e_0 y e_1 son conocidos como epipolos.

Los epipolos son la proyección del centro óptico de una de las cámaras sobre el plano de proyección de la otra cámara.

Por otro lado, la recta que queda definida por la proyección del punto *P* sobre el plano de una cámara y el epipolo correspondiente a esa cámara se define como línea epipolar. Para cada cámara habrá una línea epipolar. Por ejemplo, la línea epipolar de la cámara situada a la izquierda será la recta comprendida por los puntos p_0 y e_0.

De acuerdo con la geometría descrita, se cumple que toda línea epipolar de una imagen atraviesa el epipolo de la misma.

Por último, el plano epipolar es aquel que está formado por el punto *P* y los epipolos e_0 y e_1, o lo que es lo mismo, es el plano definido por el punto *P* y los centros ópticos O_0 y O_1.

Atendiendo a la geometría epipolar, se pueden realizar una serie de afirmaciones:

- Todo punto situado dentro del campo de visión de las cámaras, se encuentra reflejado en el plano epipolar, lo que implica que tendrá una proyección en la línea epipolar de cada cámara.

- Un punto situado en el plano proyectivo de una de las cámaras tendrá una proyección asociada de la otra imagen que se situará a lo largo de la línea epipolar de la misma. Esta imposición es conocida como restricción epipolar.

- La restricción epipolar permite que, una vez se conozca la geometría epipolar del sistema, se pueda llevar a cabo una búsqueda unidimensional de los pares de proyecciones a lo largo de las líneas epipolares, lo cual disminuye considerablemente la carga computacional al realizar búsquedas de correspondencias, a la par que se descartan puntos que puedan producir falsos positivos.

- Cualquier plano epipolar asociado a un punto del espacio intersectará siempre con la recta que une los centros ópticos de las cámaras izquierda y derecha.

- Las proyecciones de los puntos en las imágenes de las cámaras izquierda y derecha tienen el mismo orden cuando los puntos son vistos por ambas cámaras simultáneamente.

2.2.3. Matrices esencial y fundamental

La forma más común de relacionar matemáticamente los términos de la geometría epipolar, como por ejemplo la correspondencia de las dos proyecciones de un mismo punto sobre los planos de las cámaras entre sí, es por medio de las matrices esencial y fundamental.

Dicho de otra forma, al realizar la transformación de calibración para cada una de las dos proyecciones de un punto cualquiera, lo que relaciona el punto calibrado de una imagen con el punto calibrado de la otra imagen es la composición de la traslación y la rotación de una cámara con respecto a la otra.

Por ello, podemos considerar a la matriz fundamental como la matriz que junta dos proyecciones, es decir la que vincula las dos imágenes de una misma escena, mientras que la matriz esencial relaciona el movimiento relativo para llegar de una vista a la otra.

Matricialmente la relación entre cada matriz y los puntos correspondientes, viene reflejada por la fórmula:

$$p_1^T \, E \, p_0 = 0 \qquad (2.30)$$

$$q_1^T \, F \, q_0 = 0 \qquad (2.31)$$

Donde, *E* se identifica con la matriz esencial. *F* con la matriz fundamental. p_0 y p_1 son los puntos proyectados en el plano de la imagen en coordenadas espaciales, y q_0 y q_1 son dos puntos pertenecientes a un mismo plano, que vienen dados en coordenadas de píxeles.

Matriz esencial

La matriz esencial es la que proporciona la relación que existe entre las dos proyecciones de un punto cualquiera del espacio visto desde cada una de las cámaras. Es decir, vincula las coordenadas de la proyección de un punto en la cámara situada a la izquierda, con las coordenadas de la proyección del mismo punto en la cámara de la derecha.

Visto de otra forma, la matriz esencial contiene la relación física entre ambas cámaras, las cuales, en definitiva, se vinculan mediante una matriz de rotación y un vector de traslación, como se aprecia en la figura 2.13.

Por otro lado, habría que destacar una serie de propiedades de la matriz esencial:

- La matriz esencial tiene dimensiones 3 X 3, y su rango es dos. Por lo que su determinante es nulo.

- Los dos valores propios de la matriz esencial son iguales.

- La matriz esencial contiene cinco parámetros: tres correspondientes a la matriz de rotación y dos al vector de traslación.

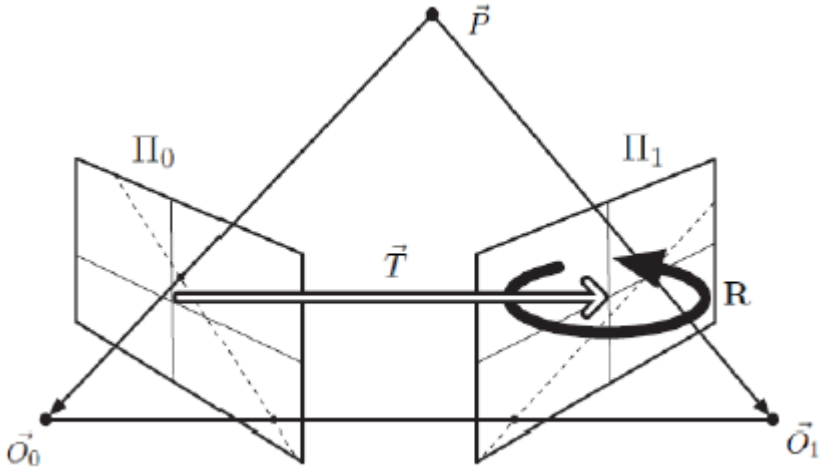


Figura 2.13. *Aplicación de la matriz esencial a una cámara*

Para poder hallar matemáticamente de una manera correcta la matriz esencial se deben tener en cuenta alguna de las deducciones anteriores, calculadas con la geometría epipolar, como se irá viendo a lo largo del posterior desarrollo de obtención de la misma.

Dado un punto cualquiera del espacio, llamado *P*, como se apreciaba en la anterior figura (2.12), se puede identificar P_0 con las coordenadas de la proyección del punto *P* sobre la cámara situada en la izquierda, O_0; y al mismo tiempo se puede considerar que P_1 son las coordenadas de la proyección del punto P sobre la cámara del lado derecho, O_1.

Interesa ahora hallar la correspondencia entre el punto P_0 y el sistema de coordenadas de la cámara de la derecha O_1, lo cual se puede lograr mediante la aplicación de la matriz de rotación *R* y del vector de traslación *T*, como se observa en la figura 2.13. En concreto, la relación queda matemáticamente reflejada por la ecuación:

$$P_1 = R(P_0 - T) \qquad (2.32)$$

Por otro lado, se sabe que en un plano definido por un vector normal *n*, y que pasa por un punto *a* cualquiera del plano, el conjunto de todos los puntos x que pertenecen al plano, deben cumplir:

$$(x - a) \cdot n = 0 \qquad (2.33)$$

Sabiendo, también, que el plano epipolar contiene al vector de traslación *T* y al vector P_0. Y que Además para crear un vector normal al plano solo hay que multiplicar vectorialmente ambos vectores.

$$n = (T\ X\ P_0) \qquad (2.34)$$

Entonces, aplicando la propiedad expuesta anteriormente en la ecuación 2.33, se tiene:

$$(P_0 - T)^T\ (T\ X\ P_0) = 0 \qquad (2.35)$$

La ecuación 2.32 se puede reescribir de la forma:

$$R^{-1} \cdot\ P_1 = (P_0 - T) \qquad (2.36)$$

Y teniendo en cuenta que cualquier matriz de rotación *R* es ortonormal, y que su traspuesta y su inversa son iguales:

$$R^{-1} = R^T \qquad (2.37)$$

La ecuación 2.35 quedaría:

$$(R^T \cdot\ P_1)^T\ (T\ X\ P_0) = 0 \qquad (2.38)$$

O lo que es lo mismo:

$$P_1^T \cdot\ R\ \cdot\ (T\ X\ P_0) = 0 \qquad (2.39)$$

Considerando que es posible redefinir el producto escalar anterior de la siguiente forma:

$$(T\ X\ P_0) = S \cdot\ P_0 \qquad (2.40)$$

Donde

$$S = \begin{pmatrix} 0 & -T_z & T_y \\ T_z & 0 & -T_x \\ -T_y & T_x & 0 \end{pmatrix} \qquad (2.41)$$

Así, la ecuación 2.39 quedaría:

$$P_1^T \cdot R \cdot S \cdot P_0 = 0 \qquad (2.42)$$

Adoptando:

$$E = R\,S \qquad (2.43)$$

Y transformando los puntos P_0 y P_1 del espacio tridimensional a coordenadas métricas localizadas en los planos de las imágenes p_0 y p_1, según las ecuaciones:

$$p_0 = \frac{f_0}{z_0} P_0 \qquad (2.44)$$

$$p_1 = \frac{f_1}{z_1} P_1 \qquad (2.45)$$

Se obtiene la expresión mostrada en la ecuación 2.30 que relaciona los mismos puntos en las dos imágenes de ambas cámaras:

$$p_1^T\, E\, p_0 = 0 \qquad (2.46)$$

Matriz fundamental

La matriz fundamental, al igual que la matriz esencial, relaciona las coordenadas de las dos proyecciones de un punto cualquiera del espacio sobre cada una de las cámaras entre sí, pero, a diferencia de la matriz esencial, lo hace en coordenadas de píxeles.

Otra característica importante de la matriz fundamental es que contiene en su interior información relevante a los parámetros intrínsecos de las cámaras que componen el sistema.

En el caso de la matriz fundamental, al igual que en el de la matriz esencial, también se deben mencionar algunas de las propiedades que cumple:

- La matriz fundamental tiene dimensiones 3 X 3, y su rango es dos. Por lo que su determinante es nulo.

- La matriz fundamental es homogénea, es decir, al ser multiplicada por un número constante no nulo el resultado continúa siendo la matriz fundamental.

- La matriz fundamental tiene siete grados de libertad: dos correspondientes a cada uno de los dos epipolos y tres vinculados al homógrafo que relaciona los dos planos de imagen.

Para poder seguir el desarrollo matemático que permite la deducción de la matriz fundamental, hay que prestar atención a la nomenclatura que se ha utilizado.

Sea q_0 un punto de la imagen izquierda situado sobre la línea epipolar l_0, según se comentó en el apartado referente a la geometría epipolar, toda línea epipolar de una imagen atraviesa el epipolo de la misma. Por lo tanto, no se comete ningún error al definir l_0 como:

$$l_0 = e_0 x q_0 = [e_0]_x \, q_0 \qquad (2.47)$$

Donde e_o es el epipolo de la cámara de la izquierda. Y se puede representar de la siguiente forma:

$$[e_0]_x = \begin{pmatrix} 0 & -e_x & e_y \\ e_y & 0 & -e_x \\ -e_y & e_x & 0 \end{pmatrix} \qquad (2.48)$$

Teniendo en cuenta que:

$$l_1 \simeq P_1^{-1} P_0^T \, l_0 \qquad (2.49)$$

Se puede sustituir en la ecuación anterior, quedando:

$$l_1 \simeq P_1^{-1} P_0^T [e_0]_x q_0 \qquad (2.50)$$

Por otro lado, al estar q_1 situado en la línea epipolar l_1:

$$q_1^T l_1 = 0 \qquad (2.51)$$

Si se define la matriz fundamental como:

$$F = P_1^{-1} P_0^T [e_0]_x \qquad (2.52)$$

Y se sustituye la ecuación (2.50) en la (2.51), se obtiene la expresión definitiva mostrada en la ecuación 2.31 que relaciona las dos imágenes en coordenadas de píxeles:

$$q_1^T F q_0 = 0 \qquad (2.53)$$

Además de las propiedades algebraicas mencionadas anteriormente, la matriz fundamental también tiene un conjunto de propiedades geométricas, que se pueden deducir del desarrollo descrito:

- Dada una geometría bifocal, la matriz fundamental es función exclusiva de esta geometría. Lo cual significa que es independiente de todos los puntos *P* de la escena y de las proyecciones p_1 y p_2 de los puntos de la escena.

- La matriz fundamental define las líneas epipolares. Se pueden calcular las líneas epipolares con la matriz fundamental mediante las siguientes ecuaciones:

$$l_1 = F q_0 \qquad (2.54)$$

$$l_0 = F^T q_1 \qquad (2.55)$$

- La matriz fundamental define los epipolos. Se pueden calcular los epipolos a partir de la matriz fundamental mediante las siguientes ecuaciones:

$$F\ e_0 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \qquad (2.56)$$

$$F^T\ e_1 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \qquad (2.57)$$

Por último, cabe destacar la relación matemática existente entre las matrices esencial y fundamental:

$$F = (K_1^{-1})^T\ E\ K_0^{-1} \qquad (2.58)$$

Donde K_0^{-1} y K_1^{-1} son las matrices que contienen los parámetros intrínsecos de calibración de las cámaras izquierda y derecha respectivamente, como se comentó en el apartado 2.1.2.

Respecto al cálculo final de la matriz fundamental, existen dos enfoques posibles. El primero es un planteamiento rápido y muy eficiente que requiere del conocimiento de las matrices de proyección, desconocidas en la mayor parte de las ocasiones (salvo que sea un entorno fuertemente controlado), lo que descarta su utilización en el caso del presente proyecto.

El segundo planteamiento, que se implementa con más generalidad que el primero, no exige el conocimiento de las matrices de proyección. Este enfoque se basa en la correspondencia de la proyección de los puntos de la imagen izquierda con sus homólogos de la derecha, y en el uso de la ecuación 2.53.

Debido a las propiedades de la matriz fundamental, con ocho pares de puntos se podrían aplicar varios de los algoritmos que resuelven el segundo planteamiento para obtener la propia matriz fundamental.

De entre todos ellos, para su cálculo en el proyecto se ha utilizado, como se verá en el apartado correspondiente, un método fundamentado en el algoritmo de RANSAC. El cual utiliza la función de pseudogradiente para minimizar el error cuadrático medio de las ocho incógnitas que se corresponden cada una con uno de los pares de puntos elegidos.

2.2.4. Calibración estéreo

De la misma manera que la calibración de una cámara simple se podía simplificar, en resumidas cuentas, en la obtención de los parámetros intrínsecos y extrínsecos de la propia cámara, la calibración estéreo trata de determinar la relación geométrica entre dos cámaras en el espacio.

La forma de representar dicha relación entre las cámaras es mediante la matriz de rotación *R* y el vector de traslación *T*. Por tanto, el objetivo principal de la calibración estéreo es la estimación de los elementos que componen tanto la matriz *R* como el vector *T*.

Para lograr calcular estos parámetros, se tiene que empezar por identificar la posición de un mismo punto *P* en ambas cámaras.

Una vez localizado, se puede mover el punto *P* al sistema de coordenadas de cada una de las cámaras utilizando una de las expresiones siguientes, según se quiera referenciar al sistema de coordenadas de la cámara de la izquierda o al de la cámara de la derecha:

$$P_0 = R_0 P + T_0 \qquad (2.59)$$

$$P_1 = R_1 P + T_1 \qquad (2.60)$$

Donde *P* es un punto arbitrario, antes mencionado, perteneciente al sistema de coordenadas global. P_0 y P_1 son las coordenadas físicas del punto *P* referenciadas a cada una de las cámaras, es decir, cuyo origen coordenado está situado en la cámara izquierda o derecha respectivamente. R_0 y R_1 son las matrices de rotación que permiten describir a los sistemas de las cámaras izquierda y derecha, en función de un sistema de coordenadas global. Y T_0 y T_1 son los vectores de traslación que complementan a las matrices de rotación para describir los sistemas de las cámaras izquierda y derecha.

Por otro lado, las dos vistas se pueden relacionar mediante *R* y *T* según la ecuación:

$$P_0 = R P_1 + T \qquad (2.61)$$

Sustituyendo:

$$R_0 P + T_0 = R(R_1 P + T_1) + T \qquad (2.62)$$

$$(R_0 - R R_1) P + T_0 - R T_1 - T = 0 \qquad (2.63)$$

Al ser *P* un punto arbitrario cualquiera, en el caso general no será obligatoriamente nulo, de lo que se deducen necesariamente las dos siguientes restricciones:

$$R_0 - R\,R_1 = 0 \qquad (2.64)$$

$$T_0 - R\,T_1 - T = 0 \qquad (2.65)$$

Despejando:

$$R = R_0\,R_1^T \qquad (2.66)$$

$$T = T_0 - R\,T_1 \qquad (2.67)$$

Al disponer de R_0, R_1, T_0 y T_1 para cada una de las imágenes tomadas del patrón de calibración, se puede calcular un valor de *R* y de *T* por cada imagen, lo que proporciona varias medidas de un mismo parámetro, con lo cual, todas estos resultados han de ser similares para obtener una buena aproximación para el valor real de *R* y de *T.*

2.2.5. Rectificación estéreo

El proceso de rectificación estéreo, además de eliminar distorsiones introducidas por la cámara, tiene el objetivo principal de modificar las dos imágenes independientes de manera que parezca que hayan sido tomadas por cámaras perfectamente alineadas entre sí, ó al menos conseguir que los pares de imágenes estén alineados, haciendo coincidir las líneas epipolares de ambas imágenes, con la finalidad de facilitar la búsqueda de correspondencias, al convertirla en un problema unidimensional.

Para lograr que las imágenes estén perfectamente alineadas, han de tener el mismo plano de imagen, es decir, sus ejes principales deben interceptarse en el infinito.

Con la calibración estéreo se logra obtener la matriz de rotación *R* y el vector de traslación *T*, los cuales rotan y trasladan respectivamente la imagen de la cámara derecha hacia la imagen de la cámara izquierda.

Sin embargo, estas transformaciones geométricas no garantizan que ambas imágenes estén perfectamente alineadas, debido a que las distancias focales de las dos cámaras pueden no ser iguales. Por ello, y para garantizarlo, se utilizan, por ejemplo el algoritmo de Bouguet o el algoritmo de Hartley.

Figura 2.14. *Imágenes rectificadas*

Al aplicar alguno de los algoritmos, se consigue también obtener una estimación de la disparidad que hay entre las dos imágenes, con la que posteriormente se puede hallar la profundidad a la que se encuentran los objetos en la imagen, mediante un proceso de triangulación simple.

El algoritmo de Bouguet trata de minimizar los cambios de la reproyección, y al mismo tiempo, trata de maximizar las áreas comunes de las dos imágenes. Por otro lado, también intenta minimizar las distorsiones introducidas durante el proceso de rectificación. Para poder implementar este algoritmo es imprescindible haber calculado la matriz de rotación *R* y el vector de traslación *T* previamente con la calibración estéreo.

Por el contrario, el algoritmo de Hartley se basa en localizar homografías que transformen las imágenes, de forma que las líneas epipolares de las propias imágenes se intercepten en el infinito. Para lograrlo, el algoritmo trabaja con una serie de correspondencias entre pares de imágenes.

Las correspondencias pueden estar precargadas en el algoritmo, o bien, puede buscarlas el mismo algoritmo por su cuenta en el conjunto de imágenes que se desean rectificar.

La opción de que la búsqueda de correspondencias sea realizada por el propio algoritmo proporciona una trascendental ventaja: se puede calcular la matriz fundamental *F* sin necesidad de haberla obtenido previamente. Es decir, el algoritmo es capaz de hallar la matriz sin ninguna intervención del programador. Aunque también tiene alguna desventaja, la principal es la no utilización de los parámetros intrínsecos y extrínsecos de las cámaras y,por otro lado, tampoco calcula las matrices de proyección rectificadas, que son las que contienen los parámetros intrínsecos que provocan que las longitudes focales, los centros ópticos y los puntos principales sean iguales en ambas cámaras.

Otra forma de verlo es que el algoritmo de Hartley no tiene en cuenta la escala de los objetos que se ven representados en las imágenes. Es decir, dado un objeto cualquiera en la imagen, no se puede afirmar si es de dimensiones muy grandes y está situado muy lejos de las cámaras, o si por otra parte es de dimensiones muy pequeñas y está localizado muy próximo a las cámaras. Este problema se observa a continuación en la representación de la figura 2.15.

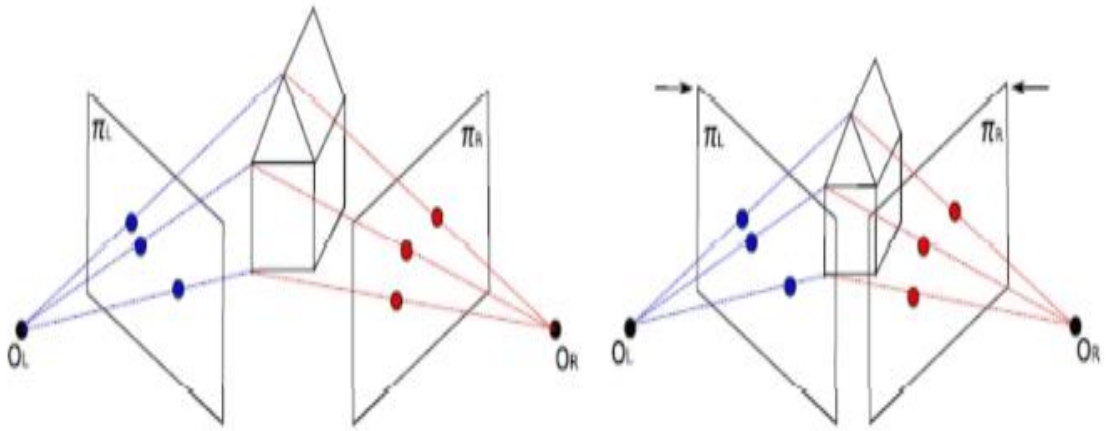


Figura 2.15. *Problema de escala en el algoritmo Hartley*

Otra desventaja a tener en cuenta es la pérdida de eficacia del algoritmo al trabajar con imágenes cuya distorsión es elevada. Esto implica que aunque mediante el algoritmo se pueda obtener la matriz fundamental sin necesidad de un proceso previo de calibración, siempre será conveniente calibrar las imágenes para eliminar en la medida de lo posible la distorsión.

A continuación se exponen con más detalle los pasos a seguir para implementar el algoritmo de Hartley, por ser el que se ha utilizado durante el desarrollo del presente proyecto.

1. Realizar la búsqueda de correspondencias entre cada par de imágenes. Serán imprescindibles un mínimo de siete pares, aunque es recomendable contar con alguno más para lograr la máxima precisión posible.

2. Obtener la matriz fundamental, si no se dispone de ella por haberla calculado en la calibración.

3. Hallar los epipolos de las imágenes. Se pueden deducir los epipolos a partir de la matriz fundamental mediante las siguientes ecuaciones:

$$F\ e_0 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \qquad (2.68)$$

$$F^T\ e_1 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \qquad (2.69)$$

4. Calcular la homografía H_R que permita mover el epipolo derecho e_1 a un punto situado en el infinito. Esta homografía dispone de diversos grados de libertad, por lo que es probable que varias de las soluciones originen imágenes distorsionadas. Para evitar dicho fenómeno, se puede definir la homografía en el entorno de un punto *P* como una multiplicación de varias matrices:

$$H_r = G\ R\ T \qquad (2.70)$$

   Donde *T* es un vector de traslación encargado de llevar el vector *P* hasta el origen del eje de abscisas. *R* es una matriz de rotación cuya función es mover el epipolo derecho e_1 al punto *(f, 0, 1)* en el eje de abscisas. Y por último, *G* es la matriz que traslada el punto *(f, 0, 1)* al infinito, y tiene la forma:

$$G = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -\frac{1}{f} & 0 & 1 \end{pmatrix} \qquad (2.71)$$

En resumen, esta homografía H_R será válida para los puntos cercanos a un punto *P* cualquiera. Y sólo se someterán a una rotación y una traslación.

5. Seleccionar la homografía H_L perteneciente a la imagen de la cámara de la izquierda de manera que se obtenga que la suma de distancias sea mínima:

$$\sum_i (H_L\, x_{Li}\, , H_R\, x_{Ri}) \qquad (2.72)$$

6. Aplicar las transformaciones a las imágenes izquierda y derecha con sus homografías correspondientes H_L y H_R. De esta forma las imágenes resultantes dispondrán de sus filas de píxeles perfectamente alineadas entre sí.

2.2.6. Búsqueda de correspondencias

La extracción de la información a partir de las dos imágenes estéreo, se realiza a través de la búsqueda de correspondencias. Es decir, para poder coger la información de interés de dos imágenes estéreo que representan el mismo entorno, es imprescindible localizar correctamente los pares conjugados de los píxeles en ambas imágenes.

Otra forma de ver la correspondencia es como el emparejamiento de las dos proyecciones de un solo punto del espacio físico en las imágenes tomadas por las dos cámaras, izquierda y derecha. Esta definición de correspondencia implica que únicamente se pueden identificar dichas proyecciones si en ambas imágenes está representado el punto original.

De lo anterior se deduce que la correspondencia se puede llevar a cabo en el área visible desde las dos cámaras. Cualquier punto situado fuera de ese rango u oculto por otro objeto localizado en un plano más cercano a las cámaras proporcionará un resultado erróneo.

Una vez terminado el proceso de rectificación de las imágenes se espera que el problema de la búsqueda de correspondencias entre las mismas se simplifique. Esto es posible gracias a que tras la rectificación se pasa de tener que realizar una búsqueda bidimensional de correspondencias a una unidimensional.

Ahora, las dos proyecciones del mismo punto del espacio físico real estarán en la misma línea horizontal. Es decir, compartirán coordenada vertical y la búsqueda habrá que centralizarla únicamente en el eje abscisas.

A la hora de solucionar el problema de la búsqueda de correspondencias, existen dos procedimientos principales para hacerlo. La primera metodología intenta hacer un análisis discreto, localizando las correspondencias en puntos concretos, mientras que la segunda busca correspondencias para todos y cada uno de los píxeles de las imágenes, basándose en la intensidad de los colores.

La principal ventaja de los métodos centrados en la intensidad es que permite hallar las correspondencias para todas las imágenes. Por el contrario, los procedimientos que utilizan puntos concretos solo son capaces de obtener buenos resultados en un conjunto de imágenes con unas características específicas determinadas.

Es evidente que no todas las regiones de una imagen poseen el mismo poder discriminante a la hora de determinar sin ambigüedad su par conjugado en la otra. Cualquier superficie rica en detalles aportará mayor cantidad de información que todas aquéllas que sean de una tonalidad uniforme, sin texturas o sin discontinuidades.

Por este motivo, dentro del enfoque discreto, se limita el proceso de correspondencia a regiones o características ricas en detalles que puedan proporcionar suficiente información como para que la ambigüedad se vea reducida al máximo y, por consiguiente, la posibilidad de que se produzcan emparejamientos erróneos sea mínima.

De esta forma, se utilizarán, preferiblemente, proyecciones de puntos pertenecientes a bordes, esquinas, y otras características del mundo físico real que permitan ser distinguidas con facilidad y sin equivocaciones con respecto al resto de píxeles de la imagen.

Por otro lado, cabe mencionar el problema de la oclusión, el cual, cuando se utilizan métodos discretos puede dificultar considerablemente la localización adecuada de la correspondencia entre los puntos afectados.

El problema de la oclusión se produce en el supuesto de que varios puntos del espacio físico real se proyecten en una de las imágenes como un solo punto, debido a que se solapen entre ellos. La información necesaria para calcular la profundidad de los objetos en este supuesto se recupera mediante la representación que los puntos tienen en la imagen perteneciente a la otra cámara. En la figura 2.16 se puede observar un ejemplo que ilustra el fenómeno de la oclusión.

Para solucionar el problema que causa la oclusión, los métodos referentes al enfoque discreto han de ser más robustos.

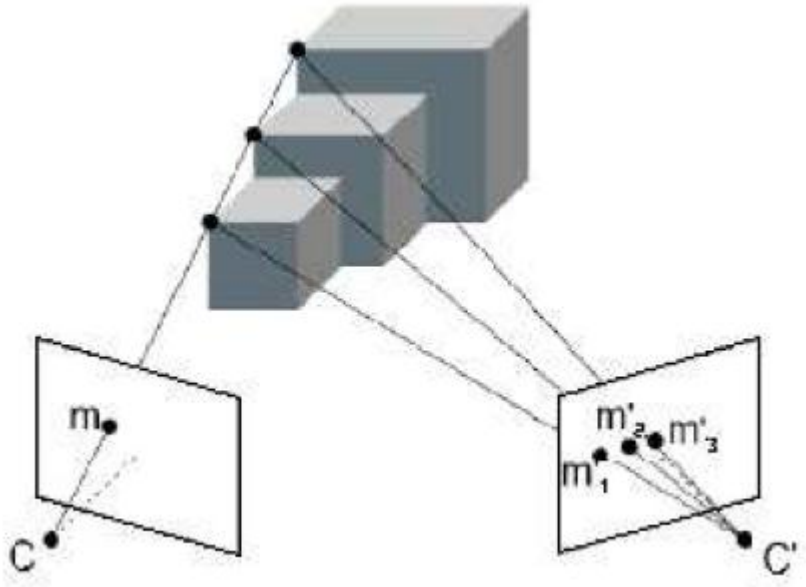


Figura 2.16. *Fenómeno de la oclusión*

Independientemente de la metodología utilizada para la búsqueda de correspondencias se pueden mencionar una serie de reglas que se cumplen en todos los casos.

En primer lugar, el orden de los elementos no se ve alterado. Es decir, se tiene que preservar la situación relativa de unos objetos con respecto a otros en ambas imágenes.

En segundo lugar, al estar las imágenes previamente rectificadas, el desplazamiento de un objeto en una imagen en función de la otra será únicamente en horizontal, en el eje de abscisas.

En tercer y último lugar, cada punto posee un solo par conjugado en la otra imagen.

Una correcta búsqueda de correspondencias facilita el cálculo posterior de la disparidad. Con ello, la determinación de la profundidad de los objetos situados en la imagen, debido a que una vez se dispone de las correspondencias la disparidad se calcula según la ecuación:

$$d(x, y) = (x_0, y_0) - (x_1, y_1) \qquad (2.73)$$

Donde x_0 e y_0 representan las coordenadas de un punto de una de las imágenes, izquierda o derecha. Por otro lado, x_1 e y_1 se identifican con las coordenadas de ese mismo punto en la otra imagen. Y por último, *d(x,y)* es la disparidad asociada a dicho punto.

2.2.7. Mapas de disparidad

La disparidad queda definida por la diferencia medida en píxeles entre la coordenada horizontal de las dos proyecciones del mismo punto una vez se ha realizado la rectificación de las mismas.

Así, las relaciones entre las coordenadas del mismo punto se ven reflejadas por las ecuaciones:

$$x_0 = x_1 + d \qquad (2.74)$$

$$y_0 = y_1 \qquad (2.75)$$

La disparidad tiene siempre un valor positivo. Por lo tanto, x_0 y x_1, que durante todo el documento hacían referencia a los valores de la coordenada x de las proyecciones de las imágenes izquierda y derecha respectivamente, en este caso se identifican con las de la imagen correspondiente tal que la disparidad adquiera un valor positivo.

Una vez obtenidos todos los valores de disparidad para los diferentes conjuntos de puntos en los que se habían calculado las correspondencias, se puede realizar un mapa de disparidad.

El mapa de disparidad es una reproducción de la imagen original en la que están representadas todas las disparidades para los diferentes puntos de la imagen. Dicho mapa de disparidad es una herramienta muy interesante y se utiliza en lo referente al cálculo de la distancia a las cámaras de los distintos objetos que se ven situados en la imagen.

La ecuación que vincula la disparidad con la profundidad a la que están los objetos es:

$$Z = \frac{f \cdot T}{d} \qquad (2.76)$$

Donde *Z* identifica la profundidad a la que se encuentra cada objeto. *T* representa la distancia, en unidades de longitud, entre los ejes ópticos de las dos cámaras, la cual no varía en ningún momento. Y *f* es la distancia focal de las cámaras, en unidades de píxeles, al igual que la disparidad.

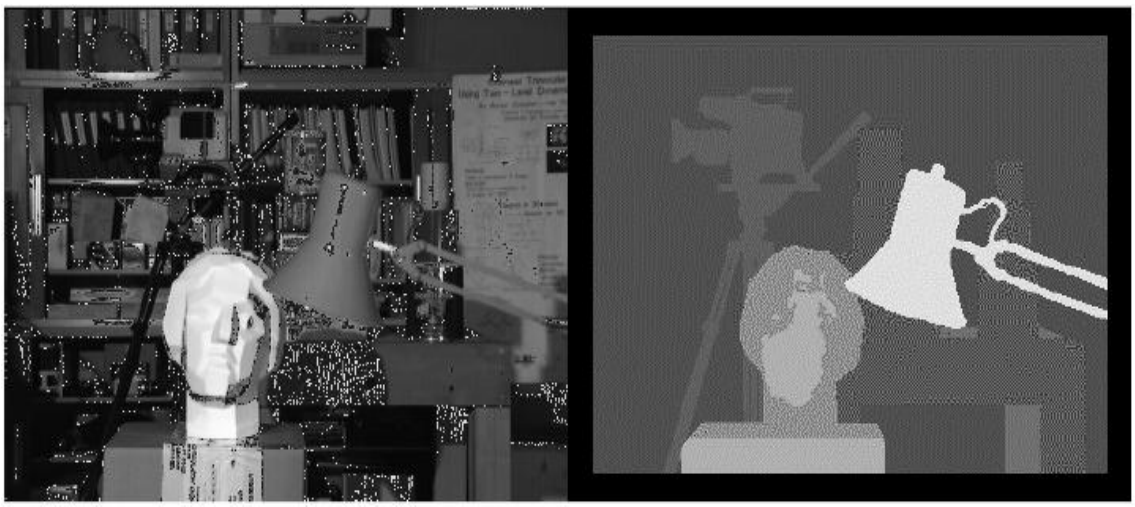


Figura 2.17. *Mapa de Disparidad*

Una propiedad a destacar del mapa de disparidades es que a medida que los objetos se alejan de las cámaras, su disparidad disminuye, como se puede apreciar en la siguiente figura:

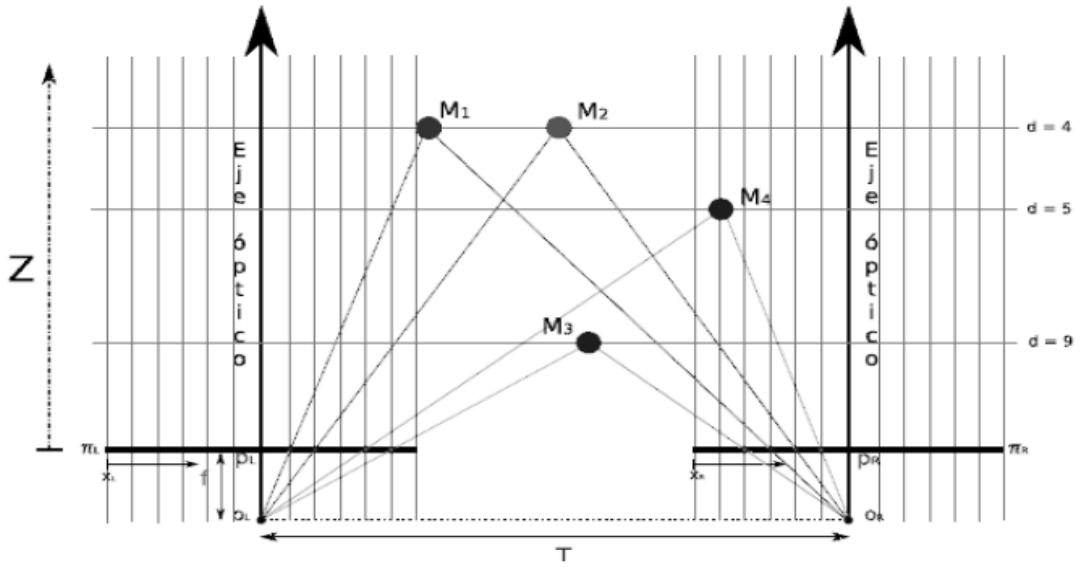


Figura 2.18. *Disparidad en función de la distancia*

2.3. OpenCV

Las librerías OpenCV son un conjunto de bibliotecas que contienen funciones orientadas principalmente al tratamiento de imágenes. La misión prioritaria de estas librerías es facilitar el diseño y la programación de aplicaciones de visión por computador en tiempo real.

El nombre OpenCV proviene de los términos anglosajones "Open Source Computer Vision Library". Sus librerías se pueden ejecutar tanto en Windows, o Linux, como en MacOS. Por otro lado, son de código abierto desarrollado en C/C++.

Es importante, a la hora de empezar a utilizar las librerías OpenCV, saber distinguir entre la nomenclatura característica de las funciones y la que poseen los distintos tipos de datos.

Para evitar confusiones la propia librería adopta una sintaxis diferente en cada caso, en lo que a funciones y tipo de datos se refiere. El problema es que ambas son muy similares y si no se presta la debida atención es frecuente caer en equivocaciones.

No obstante, no todos los tipos de datos tienen la misma nomenclatura, como se verá más detalladamente en la siguiente sección.

Por otro lado las librerías OpenCV están formadas por cinco módulos bien diferenciados, cada uno de los cuales está especializado e incluye en unas funciones concretas:

- Módulo "cv": contiene los algoritmos principales.

- Módulo "cxcore": dispone de las estructuras básicas. Es decir, las estructuras de datos y el soporte para las funciones de álgebra lineal están dentro de este módulo.

- Módulo "cvaux": incluye los algoritmos de OpenCV más experimentales, junto con las funciones auxiliares.

- Módulo "higui": contiene las funciones GUI y las relacionadas con el tratamiento de vídeos.

- Módulo "ml": es el módulo de aprendizaje automático de OpenCV.

Los cinco módulos están relacionados dentro de OpenCV según se observa en la figura 2.19.

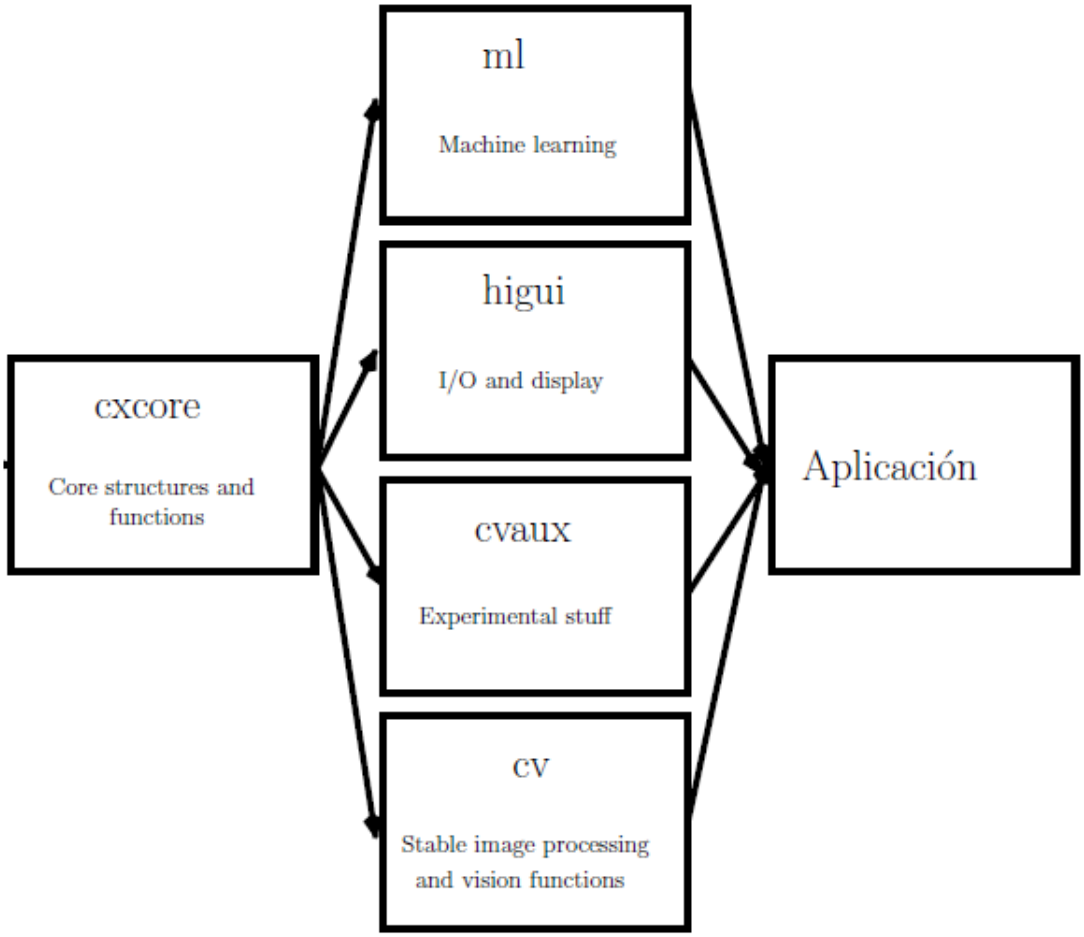


Figura 2.19. *Módulos de OpenCV*

En lo que resta de apartado se va a proceder a exponer algunos de los tipos de datos y funciones que posteriormente se utilizarán para el desarrollo del presente proyecto.

2.3.1. Datos en OpenCV

Como ya se ha comentado en el apartado anterior, antes de presentar los distintos tipos de datos que poseen las librerías, hay que hacer hincapié en la diferencia existente entre la nomenclatura de éstos y la de las funciones.

OpenCV identifica todas las funciones con el prefijo “cv”, seguidas del nombre de la función en concreto, poniendo la primera letra de cada una de las palabras que componen el nombre de dicha función en mayúscula. Por ejemplo: cvCreateImage, cvFindChessboardCorners, etc.

Sin embargo, para referirse a algunos de los tipos de datos la sintaxis es muy parecida, con la característica de que en este caso el prefijo utilizado es “Cv”. Por ejemplo: CvMat, CvPoint, etc. Mientras que otros, por el contrario, tienen su propia nomenclatura en nada parecida a la anterior, como es el caso, por ejemplo, de IplImage.

Algunos de los tipos de datos más utilizados en el proyecto se citan a continuación:

IplImage

IplImage se trata del tipo de dato básico de las librerías OpenCv. Sirve para definir a todas las imágenes independientemente de sus características individuales, las cuales se pueden modificar para cada imagen.

Se pueden ver a continuación los distintos campos que componen dicha estructura:

```
typedef struct _IplImage {
  int nSize; /* tamaño de la estructura IplImage */
  int ID; /* versión de la cabecera de la imagen */
  int nChannels; /* número de canales de color de la imagen*/
  int alphaChannel;
  int depth; /* profundidad de la imagen en píxeles; tipo de valor de los píxeles */
  char colorModel[4];
  char channelSeq[4];
  int dataOrder;
  int origin;
  int align; /* alineación de 4 u 8 bytes*/
  int width; /* anchura de la imagen en píxeles*/
  int height; /* altura de la imagen en píxeles*/
  struct _IplROI *roi; /* puntero a la ROI si existe */
  struct _IplImage *maskROI; /*puntero a la máscara ROI si existe */
  void *imageId; /* uso de la aplicación */
  struct _IplTileInfo *tileInfo; /* contains information on tiling
  int imageSize; /* tamaño útil en bytes */
  char *imageData; /* puntero a la imagen alineada */
  int widthStep; /* tamaño de alineación de cada línea de la imagen en bytes */
  int BorderMode[4]; /* modo del borde superior, inferior, izquierdo y derecho*/
  int BorderConst[4]; /* constantes para el borde superior, inferior, izquierdo y derecho*/
  char *imageDataOrigin; /* puntero a la imagen completa, sin alinear*/
} IplImage;
```

Entre los campos que durante el proyecto han sido de mayor utilidad hay que mencionar los referentes al tamaño de la imagen (*width* y *height*), así como el puntero **imageData*.

Cabe hacer especial hincapié en los campos *nChannels* y *depth*. El campo *nChannels* indica el número de canales de color de la imagen. Las imágenes representadas en escala de grises poseen un solo canal, mientras que las que lo hacen en color disponen de entre tres o cuatro.

Por su parte, el campo *depth* contiene información acerca del tipo de valor que tienen los píxeles. Los posibles valores que puede adquirir dicho campo son los siguientes:

- IPL_DEPTH_8U: Enteros sin signo de 8 bits (unsigned char)
- IPL_DEPTH_8S: Enteros con signo de 8 bits (signed char o char)
- IPL_DEPTH_16S: Enteros de 16 bits con signo (short int)
- IPL_DEPTH_32S: Enteros con signo de 32 bits (int)
- IPL_DEPTH_32F: Números en punto flotante con precisión de 32 bits (float)

CvMat

CvMat es una de las estructuras que más se utilizan a la hora de operar con imágenes. Concretamente, permite trabajar con las imágenes como si fueran matrices, almacenando un píxel en cada posición de la matriz.

Se pueden ver a continuación los distintos campos que componen dicha estructura:

```
typedef struct CvMat{
  int rows; /* número de filas*/
  int cols; /* número de columnas*/
  CvMatType type; /* tipo de matriz*/
  int step;
  union{
  float* fl; /* puntero a los datos de tipo float*/
  double* db; /* puntero a los datos de doble precisión*/
  }data;
}CvMat
```

CvSize

CvSize es una estructura cuya finalidad consiste en dar las dimensiones de un rectángulo medidas en píxeles. Una de las aplicaciones más frecuentes es definir las dimensiones de una nueva imagen.

Se pueden ver a continuación los distintos campos que componen dicha estructura:

```
typedef struct CvSize{
  int width; /* anchura del rectángulo (valor en píxeles)*/
  int height; /* altura del rectángulo (valor en píxeles)*/
}CvSize;
```

CvPoint

Los tipos de datos CvPoint y CvPoint2D23F definen las coordenadas de un punto. En el caso del primer tipo con números enteros, mientras que el segundo lo hace con número en punto flotante.

Se puede ver a continuación que la estructura CvPoint está compuesta por los campos:

```
typedef struct CvPoint{
  int x; /* coordenada x */
  int y; /* coordenada y */
}CvPoint;
```

Por otro lado, análogamente al anterior, la estructura CvPoint2D32f la componen los siguientes campos:

```
typedef struct CvPoint2D32f{
  float x; /* coordenada x */
  float y; /* coordenada y */
}CvPoint2D32f;
```

Estas estructuras, al igual que algunas de las que se han explicado previamente en este apartado, suelen estar complementadas con funciones que facilitan su uso y definición. Por ejemplo, en este último caso, estas funciones serían cvPoint y cvPoint2D32f.

2.3.2. Funciones en OpenCV

En primer lugar conviene hacer hincapié en las funciones asociadas a los tipos de datos mencionados en el apartado anterior.

cvCreateMat

La función cvCreateMat permite organizar la estructura matricial de una manera sencilla. Generalmente está vinculada con la creación de datos tipo CvMat. La utilización de esta función se identifica con la formación del encabezado de la imagen y la ubicación de los datos dentro de la matriz.

Se pueden ver a continuación los distintos campos por los que está compuesta dicha función:

CvMat* cvCreateMat(int rows, int cols, int type);
- rows: número de filas de la matriz
- cols: número de columnas de la matriz
- type: tipo de los elementos de las matrices. Se especifica de la forma:
  - CV_<bit_depth>(S|U|F)C<number_of_channels>. Siendo:
    - bit_depth: profundidad de bit (8,16,32...)
    - number of channels: el número de canales de la matriz
    - (S|U|F): el tipo de datos de bit:
      - S: con signo
      - U: sin signo
      - F: flotante

Un ejemplo aclaratorio sería: *CV_32SC2.*

cvSize

La función cvSize complementa la generación de datos tipo CvSize y sirve para definir las dimensiones de éstos.

Se pueden ver a continuación los distintos campos por los que está compuesta dicha función:

CvSize cvSize(int width, int height);
- width: anchura del rectángulo
- height: altura del rectángulo

cvPoint

Las funciones cvPoint y cvPoint2D32f son las asociadas a los tipos de datos CvPoint y CvPoint2D32f respectivamente.

Se pueden ver a continuación los distintos campos por los que están compuestas dichas funciones:

CvPoint cvPoint(int x, int y);
int x: coordenada x
int y: coordenada y

CvPoint2D32f cvPoint2D32f(double x, double y);
double x: coordenada x
double y: coordenada y

Seguidamente se citan otras funciones de aplicación frecuente a lo largo del presente proyecto.

cvNamedWindow

La función cvNamedWindow sirve para crear una ventana gráfica en la que se podrá mostrar una imagen.

Se pueden ver a continuación los distintos campos por los que está compuesta dicha función:

void cvNamedWindow(char name, int type);
name: cadena de caracteres que sirve como nombre de la ventana
type: formato de tamaño de la ventana: CV_WINDOW_AUTOSIZE (o se pondrá un “1” para seleccionar esta opción)

cvShowImage

La función cvShowImage se utiliza para representar la imagen indicada en la ventana correspondiente.

Se pueden ver a continuación los distintos parámetros por los que está compuesta dicha función:

void cvShowImage (char name, CvArr* img)
name: nombre de la ventana donde se dibujará la función
img: imagen que se desea dibujar

cvDestroyAllWindows

La función cvDestroyAllWindows elimina todas las ventanas gráficas que se hayan creado previamente. No es necesario introducir ningún parámetro en esta función.

```
void cvDestroyAllWindows();
```

cvReleaseImage

La función cvReleaseImage permite liberar el espacio de memoria que ha sido asignado previamente para guardar una imagen. Esta función dispone de un único parámetro.

```
void cvReleaseImage(& CvArr* img);
  img: es el nombre de la imagen que se desea liberar
```

A parte de las funciones ahora mencionadas, en apartados posteriores se irán definiendo y explicando otras funciones también utilizadas durante el desarrollo del proyecto.

3. Resultados

3.1. Introducción:

A lo largo del presente capítulo se va a mostrar la manera de proceder que se ha llevado a cabo para diseñar el protocolo de calibración objeto principal de este proyecto.

Junto al procedimiento utilizado, también en este apartado, se van a señalar los resultados prioritarios y más relevantes que se han ido obteniendo durante el desarrollo del protocolo indicado.

Para poder visualizar mejor los contenidos mencionados en los párrafos posteriores, a continuación se presenta un diagrama de bloques aclaratorio que engloba al sistema completo.

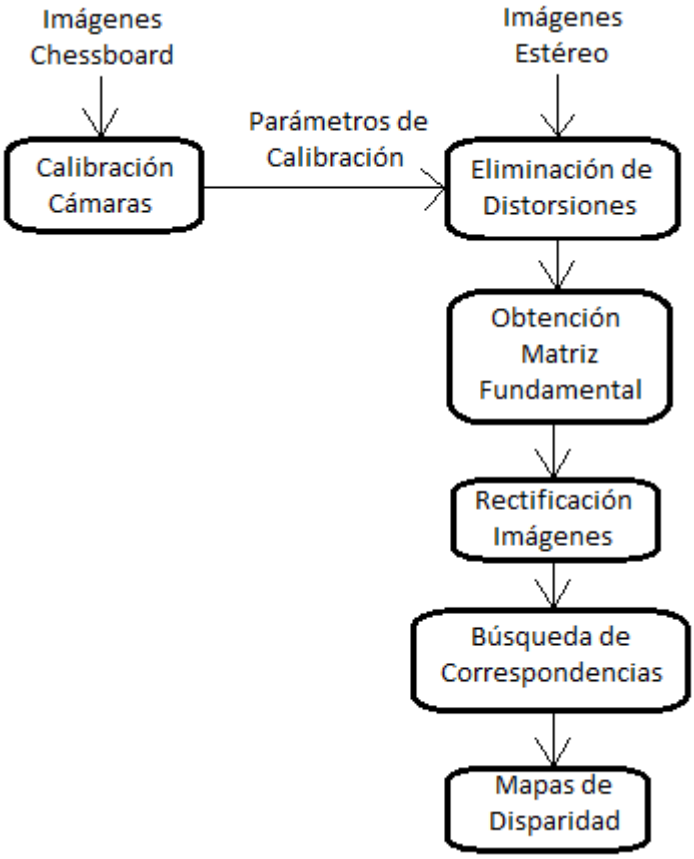


Figura 3.1. *Diagrama de bloques del sistema*

El protocolo de calibración estudiado consta de los pasos que se ven en el diagrama de bloques presentado. Todos ellos serán explicados convenientemente a lo largo del presente capítulo.

La idea general de dicho protocolo está basada en el hecho de que para realizar una fotografía estéreo es necesaria, al menos, la toma de dos imágenes del mismo paisaje. Dichas imágenes, sin ser tratadas previamente, no tienen las condiciones idóneas para ser procesadas correctamente, es decir, sin peligro de cometer ningún tipo de error.

Con el protocolo planteado se busca modificar las imágenes de partida, de tal forma que al finalizar todos los pasos satisfactoriamente se disponga de una imagen estéreo. Es decir, se conozca la profundidad a la que se encuentran todos los objetos que aparecen en la imagen.

Una vez se ha logrado obtener la profundidad de los objetos de la imagen, con los parámetros que se han ido calculando a medida que avanza el protocolo, se puede calibrar la cámara estéreo correctamente. Lo único que habría que hacer para realizar dicha calibración sería incluir estos parámetros en el propio diseño interno de la cámara.

3.2. Calibración:

El primero de todos los pasos a llevar a cabo es la obtención de los parámetros intrínsecos de las cámaras que se van a utilizar, así como también el cálculo de los coeficientes de distorsión de las mismas.

Todo ello compone el proceso de calibración, como se explicó, al principio del documento, en el apartado teórico.

Esta primera etapa de calibración se va realizar independientemente del resto de la aplicación. Se usarán unas imágenes específicas para conseguir los parámetros deseados. Una vez se disponga de dichos parámetros estas imágenes dejarán de tener repercusión en el programa.

Después de finalizar el proceso de calibración, no será necesario volver a efectuarlo, siempre y cuando no se sustituya la cámara con la que se están obteniendo las fotografías. De igual modo, habrá que calibrar cada una de las cámaras por separado.

Esta etapa comienza con la captura de un número suficiente de imágenes de un tablero de ajedrez (chessboard), mínimo ocho, con distintas orientaciones. Cuanto mayor sea la variabilidad en las rotaciones de las imágenes, será mejor para la correcta obtención de los parámetros de la cámara.

En la figura 3.2 se puede observar una secuencia ideal de imágenes que englobaría la mayor parte de las orientaciones posibles del tablero de ajedrez para la posterior calibración.

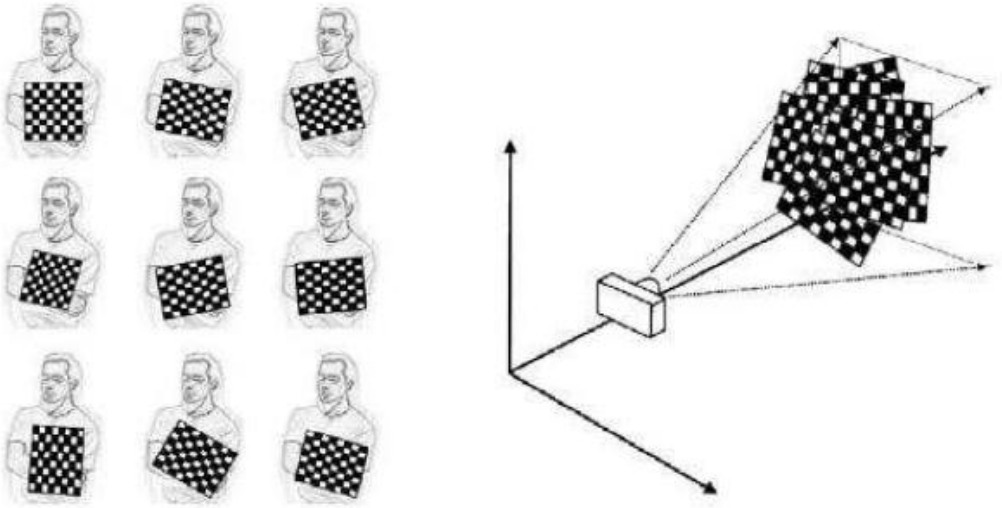


Figura 3.2. *Orientación ideal de las imágenes*

Una de las cuestiones más importantes de las capturas es que todas las esquinas interiores del tablero se vean representadas en cada imagen. Esta restricción es debida a que el algoritmo que se va a implementar localiza un número prefijado de esquinas, según las que tenga el patrón de calibración.

En caso de no encontrar todas las esquinas interiores se produciría un error en la función de búsqueda y la imagen o imágenes incompletas no se tendrían en cuenta a la hora de calcular los parámetros deseados.

Todo ello se lleva a cabo de manera sencilla gracias a la aportación de las librerías de OpenCv. Para garantizar su correcto funcionamiento, solo hay que introducir las variables adecuadas, dado que las funciones ya están desarrolladas y, normalmente, con resultados satisfactorios.

A continuación se muestra un ejemplo de las capturas mínimas utilizadas en el proyecto para determinar los parámetros de una de las cámaras. En él, se puede observar la variabilidad de las orientaciones escogidas.

Figura 3.3. *Ejemplo del proyecto con el número mínimo de patrones*

Tras haber cargado todas las imágenes con el patrón de calibración que se van a utilizar, el programa llama a la función cvFindChessboardCorners para localizar las coordenadas de las esquinas interiores del tablero. La función devolverá el valor "1" en caso de que se hayan encontrado todas.

Dentro del programa, la llamada a la función quedaría expresada de la siguiente manera:

```
int chessboardcorners1 = cvFindChessboardCorners (iml1, chessboard1, corners1, &cornercount1, CV_CALIB_CB_ADAPTIVE_THRESH);
```

La función ha de ser usada independientemente para cada una de las imágenes capturadas.

Las variables necesarias para definir la función son las que se muestran a continuación.

iml1: puntero a una matriz compuesta por la imagen fuente, en escala de grises, de la que se busca obtener las esquinas interiores del patrón

chessboard1: número de esquinas interiores del tablero en cada dirección, filas y columnas

corners1: vector de salida con las coordenadas de las esquinas detectadas

&cornercount1: número de esquinas encontradas

CV_CALIB_CB_ADAPTIVE_THRESH: umbral de intensidad

En este caso el criterio con el que se ha calibrado el tablero es de umbral de intensidad. En la tabla siguiente se muestran los distintos tipos de calibración disponibles en la función.

Nombre del método	Descripción
CV_CALIB_CB_DEFAULT	Método usado por defecto
CV_CALIB_CB_ADAPTIVE_THRESH	Umbral de intensidad
CV_CALIB_CB_NORMALIZE_IMAGE	Normaliza la imagen antes de utilizar otro método
CV_CALIB_CB_FILTER_QUADS	Se basa en otro criterio adicional como el perímetro o el área

Tabla 3.1. *Criterios de búsqueda de esquinas en el tablero*

Como se puede observar, no todos los criterios son excluyentes. Otra opción podría ser simultanear por ejemplo CV_CALIB_CB_NORMALIZE_IMAGE junto con el que se ha implementado a la hora de desarrollar el proyecto, CV_CALIB_CB_ADAPTIVE_THRESH.

Seguidamente, en el programa se utiliza la función cvFindCornerSubPix, cuya finalidad es buscar una mayor precisión en la localización de las coordenadas de las esquinas del tablero. Lo cual se realiza mediante un método iterativo.

```
cvFindCornerSubPix( iml1, corners1, cornercount1, cvSize(7,11), cvSize(-1, 1),
cvTermCriteria( CV_TERMCRIT_ITER,30,0.1));
```

Los parámetros que integran dicha función en el desarrollo del programa, quedarían:

iml1: imagen fuente en la que se van a tratar de hallar las esquinas

corners1: vector de entrada con las coordenadas de la primera iteración y vector de salida con las coordenadas definitivas

cornercount1: número de esquinas encontradas

cvSize(7,11): dimensiones del tablero en cada dirección, filas y columnas

cvSize(-1,-1): área del rectángulo en el que se efectúa la búsqueda dentro de la imagen. En este caso, se realiza en toda ella

cvTermCriteria(CV_TERMCRIT_ITER,30,0.1): criterio de finalización de las iteraciones

Un método de comprobación para averiguar si las esquinas han sido localizadas correctamente, es mediante la función cvDrawChessboardCorners. La cual imprime sobre la imagen del tablero de ajedrez original, la situación de las esquinas que se han hallado tras la utilización de la función cvFindChessboardCorners .

Esta comprobación habría de ser llevada a cabo por separado para todas las imágenes de tablero de ajedrez que se considere oportuno.

La propia función no muestra ninguna pantalla emergente. Para imprimir los resultados y así poder ver gráficamente la localización estimada, habría que llamar a la función cvShowImage, explicada en el capítulo anterior.

La función cvDrawChessboardCorners dentro del programa junto con los parámetros que la componen, quedarían de este modo:

cvDrawChessboardCorners(iml1,chessboard1,corners1,60,chessboardcorners1);

iml1: imagen de destino sobre la que se van a marcar las esquinas

chessboard1: número de esquina interiores del tablero, filas y columnas

corners1: vector de salida con las coordenadas de las esquinas detectadas

60: número total de esquinas a detectar

chessboardcorners1: indica si todas las esquinas han sido detectadas correctamente

En caso de obtener todas las esquinas interiores del tablero la representación difiere con respecto a un resultado negativo. Se puede ver un ejemplo gráfico en las figuras 3.3 y 3.4.

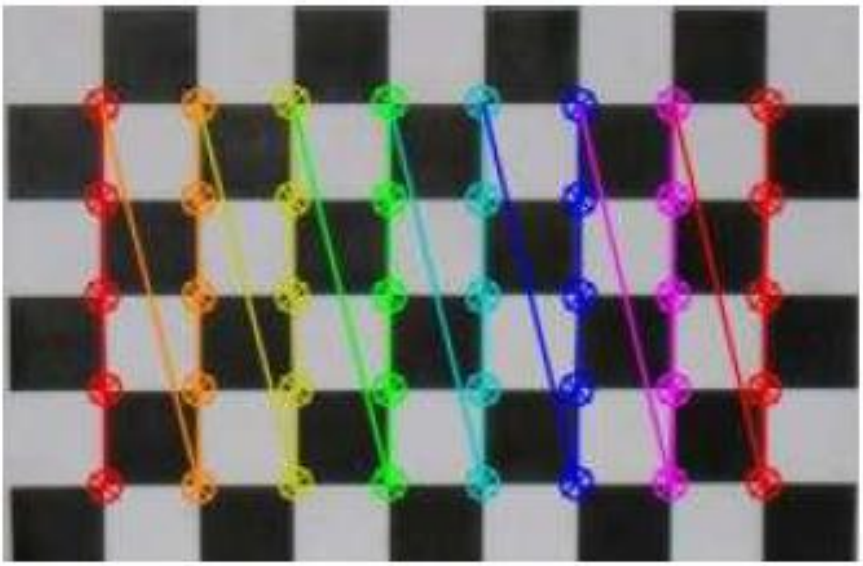


Figura 3.4. *cvDrawChessboardCorners con todas las esquinas bien localizadas*

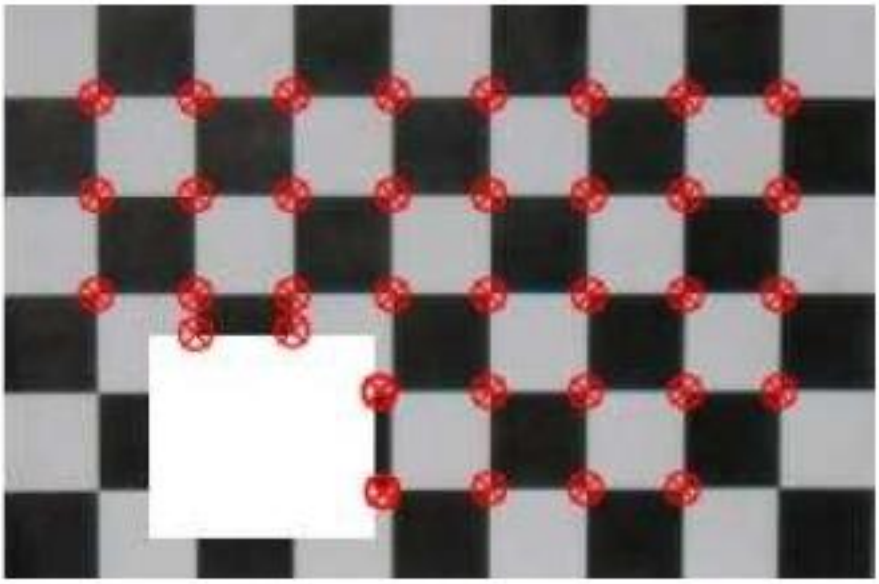


Figura 3.5. *cvDrawChessboardCorners sin localizar todas las esquinas*

Para finalizar el proceso de calibración, y una vez se hayan localizado todas las esquinas requeridas, con un mínimo de ocho imágenes patrón, se hace una llamada a la función cvCalibrateCamera2. Esta función permite calcular los parámetros intrínsecos de la cámara, así como sus coeficientes de distorsión. Además los almacena en unas matrices creadas para ello.

Dicha función dentro del programa junto con los parámetros que la componen, quedarían así:

cvCalibrateCamera2 (objectpointscalib, imagepointscalib, pointcountscalib, cvSize(imag1->width/3, imag1->height), &intrinsicmatrix, &distortioncoeffs, NULL, NULL, CV_CALIB_FIX_ASPECT_RATIO);

objectpointscalib: matriz con información acerca de las esquinas halladas en los tableros. Engloba todas las imágenes utilizadas

imagepointscalib: matriz con información acerca de las esquinas halladas en los tableros. Engloba todas las imágenes utilizadas

pointcountscalib: vector que contiene el número de esquinas de cada imagen particular. Y cuya dimensión es igual al número total de imágenes

cvSize(imag1->width/3, imag1->height): dimensiones de la imagen. Se utiliza solo para inicializar la matriz intrínseca de la cámara

&intrinsicmatrix: matriz de salida con los parámetros de la cámara

&distortioncoeffs: vector de salida con los coeficientes de distorsión

CV_CALIB_FIX_ASPECT_RATIO: indica el modo de inicializar la matriz con los parámetros de la cámara

Como se puede observar las salidas que proporciona el proceso de calibración son una matriz que contiene los parámetros intrínsecos de la cámara y un vector con los coeficientes de distorsión.

Los resultados prácticos que se han calculado con el programa durante el proyecto son:

$$K_l = \begin{pmatrix} 106424 & 0 & 406 \\ 0 & 106424 & 178 \\ 0 & 0 & 1 \end{pmatrix} \qquad (3.1.)$$

$$D_l = \begin{pmatrix} -8640 \\ 69611 \\ 2 \\ -5 \\ 177 \end{pmatrix} \qquad (3.2.)$$

$$K_r = \begin{pmatrix} 34628 & 0 & 253 \\ 0 & 34628 & 169 \\ 0 & 0 & 1 \end{pmatrix} \qquad (3.3.)$$

$$D_r = \begin{pmatrix} -1103 \\ 24016 \\ 2 \\ 4 \\ 71 \end{pmatrix} \qquad (3.4.)$$

Donde K_l y D_l hacen referencia a la matriz intrínseca y al vector de distorsión de la cámara situada más a la izquierda, mientras que K_r y D_r lo hacen a la que se localizada en el lado derecho.

3.3. Distorsiones:

El proceso de calibración realizado en el apartado anterior, se debe llevar a cabo independientemente del resto del programa.

Una vez completado el mismo, se puede utilizar los resultados obtenidos para proseguir con la elaboración del protocolo.

Las imágenes con las que se va a trabajar poseen distorsiones debidas a las cámaras. En primer lugar, habrá que eliminar dichas distorsiones para operar con las imágenes en perfectas condiciones.

Las dos imágenes proceden una de cada cámara, por lo que serán necesarios tanto las matrices intrínsecas de las dos cámaras, como sus vectores de coeficientes de distorsión.

Para eliminar la distorsión de las imágenes se puede utilizar la función cvUnDistortOnce. El desarrollo de esta función en el proyecto se puede observar a continuación.

cvUnDistortOnce (imrdis, imr, *_intrinsicmatrixr, *_distortioncoeffsr, 1);

imrdis: imagen original, con distorsión

imr: imagen de destino, sin distorsión

*_intrinsicmatrixr: matriz con los parámetros intrínsecos de la cámara

*_distortioncoeffsr: vector con los coeficientes de distorsión

Seguidamente se muestran las imágenes usadas durante el proyecto, con y sin distorsión.

Figura 3.6. *Imagen derecha con y sin distorsión*

Figura 3.7. *Imagen izquierda con y sin distorsión*

Con las nuevas imágenes sin distorsión ya se estaría en disposición de calcular la matriz fundamental.

3.4. Matriz fundamental:

Para calcular la matriz fundamental las librerías de OpenCv cuentan con la función cvFindFundamentalMat(), la cual es de respuesta rápida y uso sencillo.

Para poder utilizar dicha función se requieren algunos parámetros previos, los cuales serán necesarios a la hora de implementarla, como se verá más adelante cuando se expliquen las variables que componen la función, en este mismo apartado.

A continuación, se va a realizar un emparejamiento de los puntos de la imagen izquierda con sus homólogos en la imagen derecha, el cual será de máxima importancia para lograr la matriz fundamental.

Debido a la complejidad de vincular el mismo punto de las dos imágenes entre sí, esta etapa se va a llevar a cabo mediante la modificación de una solución, en forma de ejemplo, proporcionada en las librerías OpenCv.

Las principales funciones que componen este módulo para conseguir el emparejamiento de los puntos y sus características más importantes se irán viendo secuencialmente.

En primer lugar, para poder avanzar con el programa, se han de obtener los descriptores y los puntos de especial interés de las imágenes. Este proceso se tiene que realizar en cada imagen de forma independiente.

La función encargada de conseguirlos es cvExtractSURF(), cuyos parámetros más representativos son los siguientes.

cvExtractSURF(iml,0,&objectKeypoints,&objectDescriptors,storage, parametros);

iml: imagen fuente en escala de grises en la que se van a buscar los puntos de interés

0: zona de búsqueda de puntos de interés. En este caso se realizará en toda la imagen

&objectKeypoints: puntero de salida que señala a las direcciones de los puntos de interés

&objectDescirptors: puntero de salida que señala a las direcciones de los descriptores de la imagen

storage: zona de la memoria reservada para almacenar la información de los descriptores y de los puntos de interés

parametros: indica las características para distinguir puntos de interés y descriptores. Explicada a continuación

CvSURFParams parametros = cvSURFParams(500, 1);

500: valor umbral mínimo para que los puntos de interés detectados sean válidos

1: variable boolena. Si es verdadera significa que cada imagen puede tener 128 descriptores

Una vez terminada la extracción de puntos de interés y descriptores de las dos imágenes a relacionar, se puede realizar el emparejamiento de los puntos homólogos propiamente dicho.

Para ello se utiliza la función *flannFindPairs(),* la cual dará como resultado un vector con los puntos correspondientes emparejados.

La función *flannFindPairs()* realmente no es una función como tal, sino que es un algoritmo compuesto por diversas funciones que se irán implementando según los requerimientos para lograr el correcto emparejamiento.

Dicho algoritmo dentro del programa junto con los parámetros que lo componen, quedarían:

flannFindPairs (objectKeypoints, objectDescriptors, imageKeypoints, imageDescriptors, ptpairs);

objectKeypoints: vector de entrada con los puntos de interés que representan la imagen izquierda

objectDescriptors: vector de entrada que contiene los descriptores de la imagen izquierda

imageKeypoints: vector de entrada con los puntos de interés que representan la imagen derecha

imageDescriptors: vector de entrada que contiene los descriptores de la imagen derecha

ptpairs: vector de salida que contiene los pares de puntos homólogos vinculados

La forma de trabajar simplificada del algoritmo es utilizar un método de búsqueda de características, como el del vecino más próximo.

Después, cuando todos los puntos de interés y todos los descriptores han sido analizados, el algoritmo compara los resultados correspondientes a la imagen izquierda con los de la imagen derecha.

Para terminar, vincula los que mayor parecido tienen en sus características en ambas imágenes, y los almacena ordenadamente en el anteriormente mencionado vector, *ptpairs*.

El resultado logrado durante el proyecto en el tema del emparejamiento de puntos se muestra en la figura 3.8. Las imágenes sobre las que se realiza carecen de distorsión.

Figura 3.8. *Emparejamiento horizontal de puntos*

La imagen anterior tiene un valor del parámetro "parametros" dentro de la función cvExtractSURF() de 500, como se pudo ver anteriormente. Antes de continuar se van a mostrar unas imágenes comparativas de cómo influye esa variable a la hora de conseguir más o menos emparejamientos, según sea el umbral mayor o menor.

Figura 3.9. *Emparejamiento horizontal de puntos (umbral 300)*

Figura 3.10. *Emparejamiento horizontal de puntos (umbral 700)*

En la comparativa se puede apreciar que a menor umbral mayor es el número de puntos emparejados. Y viceversa, es decir, a mayor umbral menor es el número de emparejamientos conseguido.

Ahora, con estos datos, ya se está en disposición de ejecutar la función cvFindFundamentalMat(), la cual proporciona un valor "1" en caso de que la matriz fundamental se calcule correctamente, y un valor nulo en caso de que dicha matriz no sea localizada.

Tal función dentro del programa junto con los parámetros que la componen, quedarían así:

fm_count = cvFindFundamentalMat (&_pt1, &_pt2, &Matriz_Fundamental, CV_FM_RANSAC, 1, 0.99, NULL);

fm_count: valor devuelto por la función que indica el número de matrices fundamentales calculadas

&_pt1: vector de puntos de la primera imagen. Las coordenadas de los puntos deben ser float

&_pt2: vector de puntos de la segunda imagen, igual que el anterior

&Matriz_Fundamental: salida con la matriz fundamental calculada

CV_FM_RANSAC: método de obtención de la matriz fundamental

1: máxima distancia desde el punto a la línea epipolar

0.99: nivel de confianza para la estimación de la matriz fundamental

Cabe hacer una rápida mención a los diversos métodos que propone OpenCv para deducir la matriz fundamental. En la siguiente tabla, se pueden ver tanto el nombre del algoritmo implementado como el número de puntos requeridos para su utilización.

Método	Puntos necesarios	Algoritmo
CV_FM_7POINT	N=7	7 – Points
CV_FM_8POINT	N>=8	8 – Points
CV_FM_LMEDS	N>=8	LMedS
CV_FM_RANSAC	N>=8	RANSAC

Tabla 3.2. *Métodos de obtención de la matriz fundamental*

Los resultados definitivos de la matriz fundamental proporcionados por el programa según el método aplicado (8-Points, LMedS y RANSAC, respectivamente), se citan a continuación.

$$F_{RANSAC} = \begin{pmatrix} 0 & -0{,}000131 & 0{,}012215 \\ 0{,}000117 & 0{,}000007 & -0{,}179163 \\ -0{,}012307 & 0{,}181388 & 1 \end{pmatrix} \quad (3.5)$$

$$F_{LMedS} = \begin{pmatrix} -0{,}000001 & -0{,}000010 & -0{,}012168 \\ -0{,}000008 & -0{,}000003 & -0{,}205923 \\ 0{,}013470 & 0{,}211328 & 1 \end{pmatrix} \quad (3.6)$$

$$F_{8Point} = \begin{pmatrix} -0{,}000013 & -0{,}000917 & 0{,}285476 \\ 0{,}000931 & 0{,}000018 & -0{,}296157 \\ -0{,}288480 & 0{,}290416 & 1 \end{pmatrix} \quad (3.7)$$

De las tres matrices expuestas se ha utilizado para proseguir el programa la resultante del método RANSAC, por facilitar una aproximación más exacta, y con ello, mejores soluciones en las funciones posteriores.

3.5. Rectificación:

La etapa de rectificación es de las más importantes dentro del protocolo, debido a que dependiendo de lo bien o mal que se lleve a cabo, así serán los resultados posteriores del mapa de disparidad.

Por otro lado, es un proceso complejo que está compuesto por diversas operaciones. En general, todas ellas se pueden realizar aplicando las funciones de OpenCv.

Ordenadas según su caracterización en el proyecto, las distintas operaciones serían:

- Obtener las matrices de homografía
- Trabajar con las matrices para eliminar posibles problemas debidos a las cámaras
- Eliminar los errores propiciados por la distorsión de las cámaras
- Rectificar las imágenes
- Eliminar la inclinación de las imágenes

La obtención de las imágenes rectificadas está basada en las librerías OpenCv, y en la utilización del algoritmo de Hartley, que ya quedó explicado en el apartado de teoría.

Para poder realizar la primera operación, la obtención de las homografías ó matrices de homografía en el programa, se ejecuta la función cvStereoRectifyUncalibrated(). Dicha función proporciona como resultado el valor “1” en caso de que las matrices de homografía de las dos imágenes se hayan podido calcular.

Esta función solo ha de ser llamada una vez para obtener las homografías de las dos imágenes con las que se está trabajando, una procedente de la cámara izquierda y otra de la derecha.

Esa función, dentro del programa junto con los parámetros que la componen, quedarían:

a = cvStereoRectifyUncalibrated (&_pt1, &_pt2, &Matriz_Fundamental, Dimension, &_Hl, &_Hr, 5);

a: valor devuelto por la función que indica si las dos matrices de homografía han sido calculadas

&_pt1: entrada con el vector que contiene los puntos de interés de la primera imagen

&_pt2: entrada con el vector que contiene los puntos de interés de la segunda imagen

&Matriz_Fundamental: entrada con la matriz fundamental

Dimension: tamaño de la imagen

&_Hl: salida con la matriz de homografía para la primera imagen

&_Hr: salida con la matriz de homografía para la segunda imagen

5: umbral mínimo para que los distintos pares de puntos computen a la hora de calcular las homografías

Los resultados prácticos de las matrices de homografía proporcionados por el programa, serían:

$$H_l = \begin{pmatrix} -0{,}138183 & 0{,}011567 & -19{,}805645 \\ 0{,}015435 & -0{,}185403 & -5{,}38294 \\ 0{,}000150 & -0{,}000013 & -0{,}227395 \end{pmatrix} \qquad (3.8)$$

$$H_r = \begin{pmatrix} 0{,}697576 & -0{,}091023 & 118{,}621353 \\ -0{,}091125 & 1{,}020368 & 24{,}271803 \\ -0{,}000191 & 0{,}000120 & 1{,}265244 \end{pmatrix} \qquad (3.9)$$

Donde H_l se refiere a la homografía obtenida de la imagen de la cámara situada en la izquierda, mientras que H_r lo hace a la que está localizada en el lado de la derecha.

El siguiente paso no requiere de la utilización de ninguna función propia de las librerías OpenCv, solo es necesario llevar a cabo unas operaciones matemáticas con las matrices.

Mediante la realización de las multiplicaciones matriciales que se exponen a continuación, en las ecuaciones 3.10 y 3.11, se eliminan posibles problemas debidos a las cámaras.

$$R_l = M_l^{-1} H_l M_l \qquad (3.10)$$

$$R_r = M_r^{-1} H_r M_r \qquad (3.11)$$

Donde R_l y R_r son las nuevas matrices con las que se continuará el proceso, correspondientes a las cámaras izquierda y derecha respectivamente. Y M_l y M_r representan a las matrices que contienen los parámetros característicos de las cámaras.

Como se puede observar, las multiplicaciones hay que efectuarlas independientemente para cada imagen, según la homografía que fue calculada previamente y las matrices de parámetros de cada cámara.

El resultado son otras dos matrices, cuyos valores numéricos exactos carecen de interés práctico.

Una vez se dispone de las nuevas matrices se puede ejecutar la función cvInitUndistortRectifyMap(), la cual permite eliminar los errores debidos a la distorsión de la cámara, dando como valores de salida dos vectores: uno con las coordenadas-x de mapeo de una imagen y otro con las coordenadas-y de mapeo de la misma imagen.

Esta función tendrá que ser llamada dos veces, es decir, independientemente para cada imagen.

La función, dentro del programa, junto con los parámetros que la componen, quedarían:

cvInitUndistortRectifyMap(&_mcl, NULL,&_Rl,&_mcl, mapxl,mapyl);

&_mcl: entrada de la matriz con los parámetros de la cámara correspondiente

NULL: entrada de la matriz con los coeficientes de distorsión

&_Rl: entrada de la matriz con los valores para la rectificación de la imagen

&_mcl: salida de la nueva matriz con los parámetros de la cámara tras aplicar la rectificación

mapxl: salida del vector correspondiente a las coordenadas-x de mapeo

mapyl: salida del vector correspondiente a las coordenadas-y de mapeo

Para aplicar los mapas de coordenadas calculados en el proceso anterior hay que ejecutar otra función, ésta es cvRemap().

Esta función efectúa una operación de mapeo sobre una imagen seleccionada, es decir, modifica la imagen inicial según los valores de los vectores de mapeo introducidos, ofreciendo una imagen de salida ya mapeada. Estos vectores de mapeo serán los obtenidos anteriormente.

La función, dentro del programa, junto con los parámetros que la componen, quedarían:

cvRemap (iml,rectificadal,mapxl,mapyl);

iml: entrada con la imagen fuente (izquierda) sobre la cual se va a efectuar la rectificación

rectificadal: salida con la nueva imagen de destino (izquierda) ya rectificada

mapxl: vector de entrada con el mapa de las coordenadas-x correspondientes a la imagen izquierda

mapyl: vector de entrada con el mapa de las coordenadas-y correspondientes a la imagen izquierda

Las imágenes que se consiguen después de esta etapa serán imágenes ya rectificadas. Es decir, un mismo punto en las dos imágenes tendrá la misma coordenada vertical, aunque la coordenada horizontal no coincidirá. A las imágenes también se les eliminará la distorsión.

Se pueden observar estos efectos en las siguientes imágenes.

Figura 3.11. *Imagen izquierda rectificada*

Figura 3.12. *Imagen derecha rectificada*

Para ver mejor las diferencias entre ambas imágenes, en la figura 3.13, se muestran las dos situadas una junto a la otra.

Figura 3.13. *Imágenes rectificadas izquierda y derecha juntas*

En la figura 3.13 se puede apreciar que los objetos de la imagen derecha tienen menor pendiente que los de la imagen izquierda. Esta característica se observa, principalmente, en la caja situada más cerca de las cámaras, o en el marco de la puerta.

Aunque también es cierto que, por estar rectificadas, un mismo punto de las dos imágenes estará en la misma línea horizontal.

Para comparar un punto en las dos imágenes, a continuación se dispone de una imagen en la que se ha realizado una búsqueda de correspondencias.

Figura 3.14. *Correspondencias entre las imágenes rectificadas*

En la figura 3.14 se ve cómo los puntos cuya correspondencia se ha calculado correctamente, tienen líneas horizontales para vincularse a su punto homólogo en la otra imagen.

Esta figura junto con el cómputo de las correspondencias no tiene mayor interés dentro del programa que el de demostrar que un mismo punto se sitúa en la misma línea horizontal después de la rectificación.

Si bien el proceso de rectificación habría concluido con la obtención de las imágenes anteriores, no así las operaciones citadas al principio del apartado para disponer de las imágenes en las condiciones adecuadas para calcular el mapa de disparidad.

La última operación a llevar a cabo sería la anulación de la inclinación de las imágenes. Ésta se compone a su vez de varios procesos.

El primer paso para volver verticales las líneas que han de serlo es aplicar un detector de bordes. La labor del detector de bordes es facilitar la pendiente de los objetos pertenecientes a la imagen.

En nuestro caso se utilizará el detector de bordes de Canny. Y se ejecutará mediante la función cvCanny(), incluida en las librerías de OpenCv.

La función cvCanny() dibuja los bordes de una imagen fuente en una copia de la misma imagen, que será la de destino. Los bordes son calculados según los valores umbrales indicados para su detección.

Tal función dentro del programa junto con los parámetros que la componen, quedarían:

cvCanny (rectificadal, cannyl, 50, 200, 3);

rectificadal: imagen de entrada fuente sobre la que se van a calcular los bordes de los objetos

cannyl: imagen de destino, copia de la anterior, en la que se van a representar los bordes localizados por la propia función

50: valor del umbral mínimo que determina la existencia de contorno. Sirve para unir puntos de contorno

200. valor del umbral máximo que determina la existencia de contorno. Sirve para encontrar los puntos iniciales de contornos fuertes

3: define el tamaño de la matriz que forma el filtro de Sobel, integrado en el propio algoritmo de Canny

Esta función calcula los bordes de una única imagen, por lo tanto deberá ser usada por separado para cada imagen de las que se quieran obtener. Es decir, se deberá llamar independientemente para la imagen rectificada izquierda y para la derecha.

Los resultados prácticos generados por el programa en este punto, son las siguientes imágenes, representadas en la próxima página, en la figura 3.15 y en la figura 3.16.

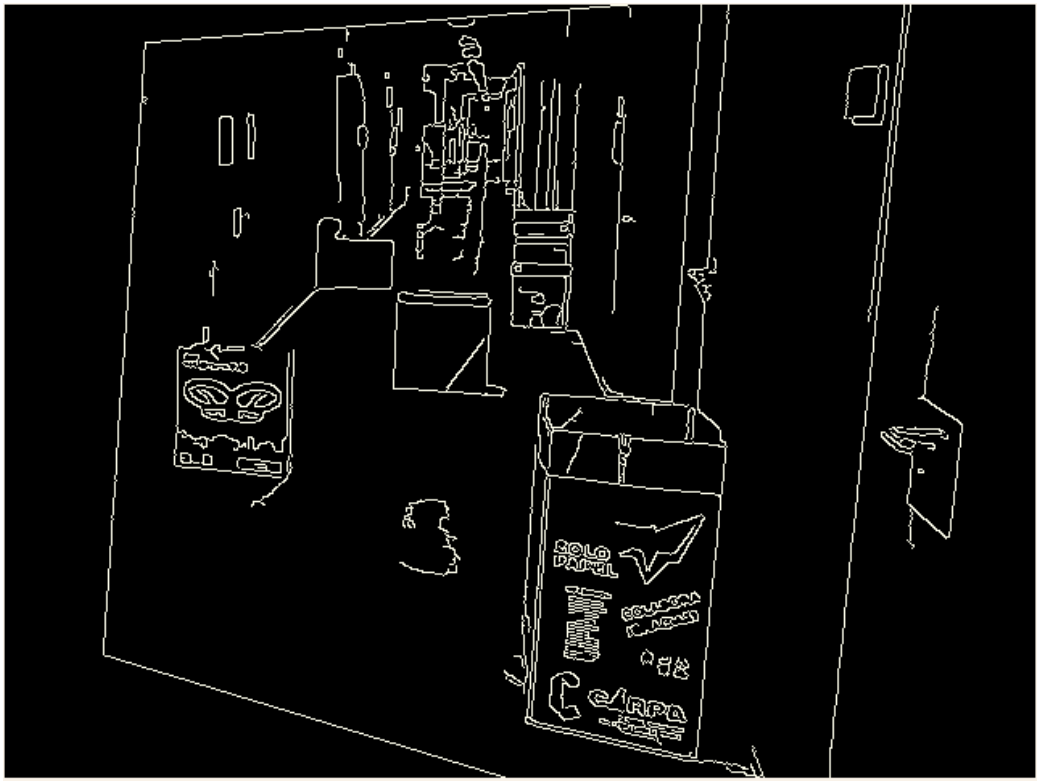


Figura 3.15. *Bordes detectados con cvCanny en la imagen izquierda*

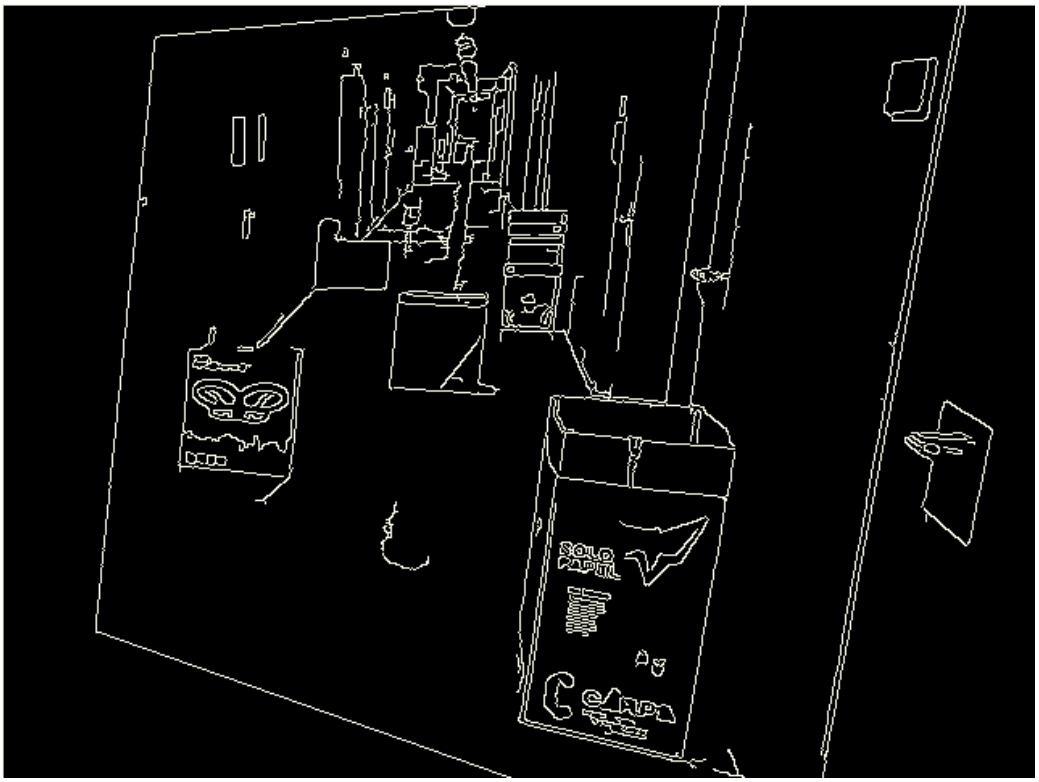


Figura 3.16. *Bordes detectados con cvCanny en la imagen derecha*

El segundo paso de esta última operación para anular la inclinación, sería la aplicación de una función para suavizar la imagen. En este caso se utilizará la función cvSmooth().

Dicha función dentro del programa junto con los parámetros que la componen, quedarían:

cvSmooth(cannyl,smoothl,CV_GAUSSIAN,3,0);

cannyl: entrada con la imagen fuente que se usará como modelo previo para suavizar

smoothl: salida con la imagen de destino, será una copia de la imagen fuente, pero ya suavizada

CV_GAUSSIAN: tipo de suavizado

3: primer parámetro para la operación de suavizado

0: segundo parámetro para la operación de suavizado. Como toma valor nulo la gaussiana, para realizar el suavizado se centra únicamente en el primer parámetro

En este caso el tipo de suavizado que se ha llevado a cabo es mediante la gaussiana. En la siguiente tabla se muestran todas las formas de suavizado que se pueden realizar con la función cvSmooth().

Tipo de Suavizado	Descripción
CV_BLUR_NO_SCALE	Realiza la suma de cada píxel con sus vecinos comprendidos entre el primer y segundo parámetro
CV_BLUR	Realiza la suma de cada píxel con sus vecinos comprendidos entre el primer y segundo parámetro. Después escala el valor por el inverso del resultado
CV_GAUSSIAN	Se realiza la transformación de la imagen mediante la gaussiana, basada en el primer y segundo parámetro
CV_MEDIAN	Encuentra la media basada en el primer parámetro y los vecinos
CV_BILATERAL	Aplica un filtro bilateral 3x3 a la imagen basado en el primer y segundo parámetro

Tabla 3.3. *Tipos de suavizado de la función cvSmooth()*

La salida que facilita esta función es una imagen suavizada según el método elegido, de los expuestos en la tabla anterior. La imagen de salida será una copia de la que se ponga como imagen fuente.

Por otro lado, esta función, al igual que la anterior, únicamente suaviza una imagen. Por lo tanto se deberá ejecutar independientemente para las dos imágenes, la izquierda y la derecha.

Las imágenes proporcionadas por el programa en este punto son las que se muestran a continuación.

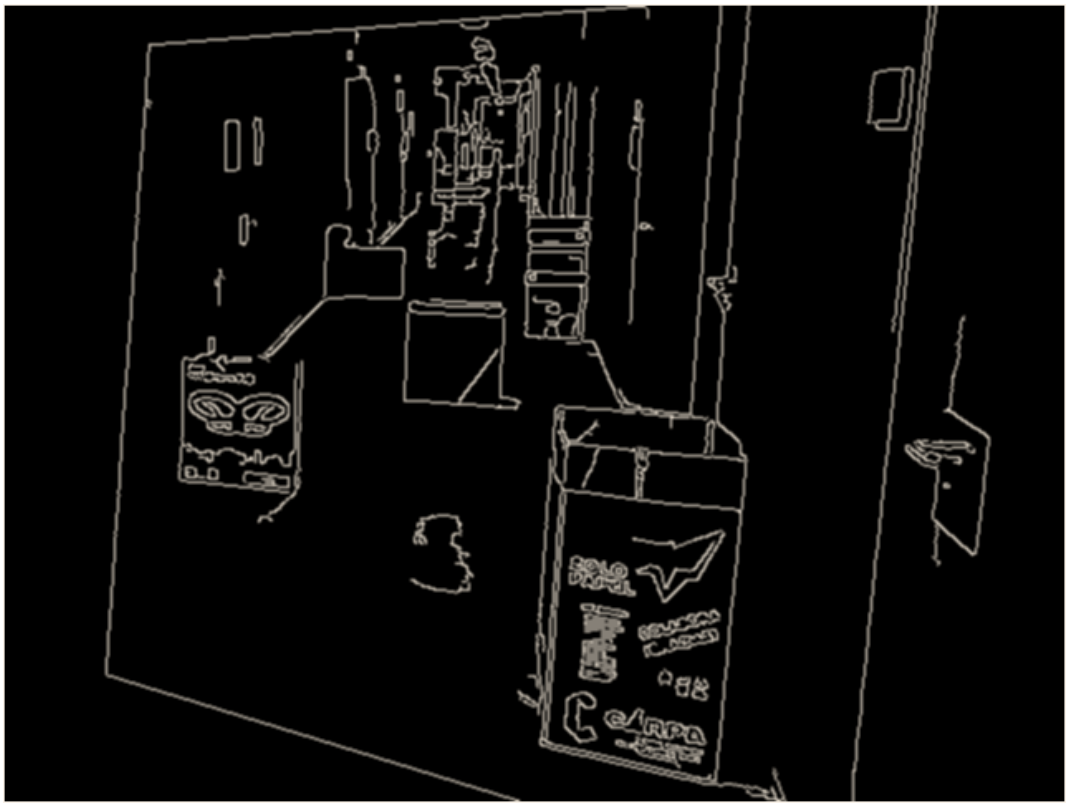


Figura 3.17. *Suavizado de bordes de la imagen izquierda*

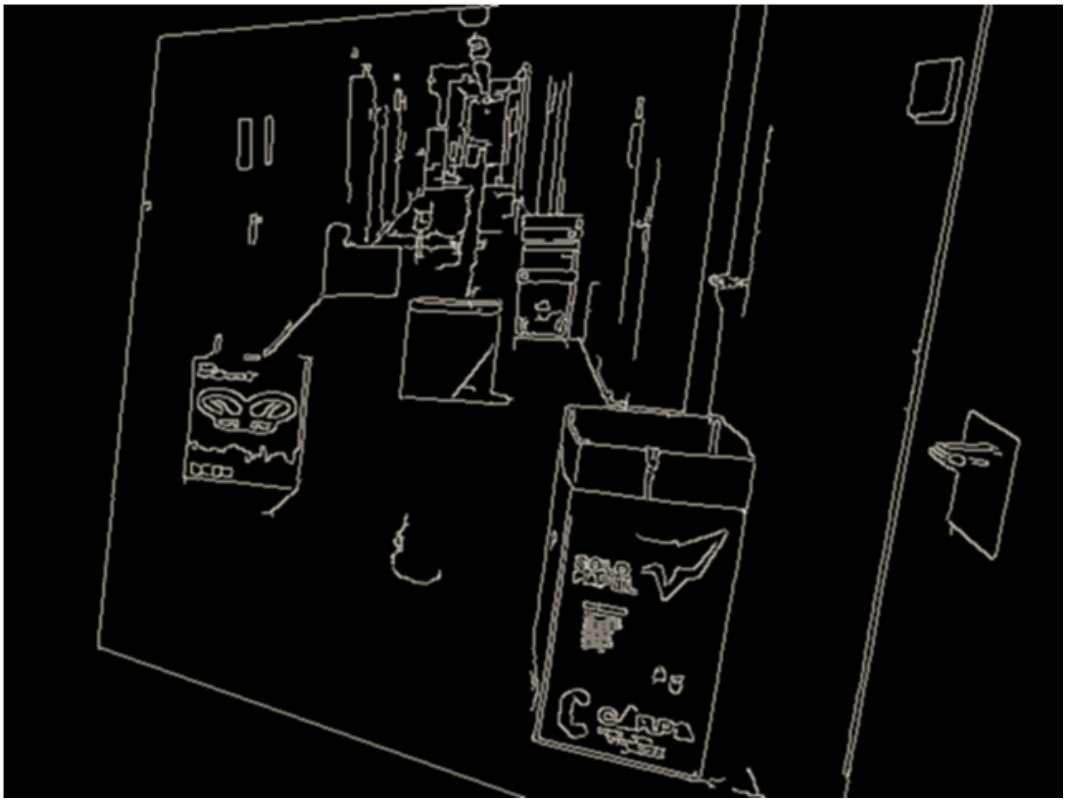


Figura 3.18. *Suavizado de bordes de la imagen derecha*

Si se realiza una comparación entre las figuras 3.15, 3.16 y 3.17, 3.18, según representen las imágenes izquierda o derecha, se aprecia que la única diferencia entre ellas es la nitidez de las líneas que simulan los objetos.

El próximo paso para lograr anular la inclinación de los objetos es identificar la pendiente de las líneas que deberían ser verticales. Este proceso se inicia con la localización de todas las líneas que hay en las imágenes.

Posteriormente se seleccionarán las que sean de interés, puesto que no se quiere poner todas las líneas verticales, solo las que deberían serlo.

Para llevar a cabo la obtención de las líneas se usará una función de las librerías OpenCv que implementa la transformada de Hough, la cual sirve para detectar las líneas que aparecen en la imagen.

Esta función es cvHoughLines2(), y proporciona un vector que contiene almacenadas todas las líneas encontradas en la imagen.

Dicha función dentro del programa junto con los parámetros que la componen, quedarían:

linesl = cvHoughLines2 (smoothl, memalmacenl, CV_HOUGH_PROBABILISTIC, 1, CV_PI/180, 80, 50, 10);

linesl: vector de salida en el que se van a almacenar todas las líneas encontradas con la función

smoothl: entrada con la imagen fuente y salida con la imagen modificada

memalmacenl: reserva de memoria requerida por la función para realizar sus modificaciones en la imagen

CV_HOUGH_PROBABILISTIC: variante de la transformada de Hough utilizada en el proceso

1: indica la resolución de la distancia medida en unidades de píxeles

CV_PI/180: resolución del ángulo medido en radianes

80: parámetro umbral. Una línea es almacenada por la función cuando su acumulador es mayor que el umbral

50: para el método utilizado este valor se refiere a la longitud mínima de una línea

10: para el método utilizado este valor indica el máximo hueco permitido entre los segmentos de una misma línea

En este caso el método que se ha usado para la detección de líneas ha sido CV_HOUGH_PROBABILISTIC. En la siguiente tabla se muestran todos los métodos disponibles en la función.

Método	Descripción
CV_HOUGH_STANDARD	Transformada de Hough clásica. Todos los puntos son representados por dos números en coma flotante
CV_HOUGH_PROBABILISTIC	Transformada de Hough probabilística, más eficaz si la imagen contiene segmentos de líneas cortos.
CV_HOUGH_MULTI_SCALE	Variante multi-escala de la transformada de Hough clásica. Las líneas son codificadas de la misma forma que en el primer método
CV_HOUGH_GRADIENT	

Tabla 3.4. *Métodos de detección de líneas*

Una vez han sido localizadas todas las líneas de la imagen se identifican las que deberían ser verticales, mediante un programa diseñado al efecto. Es decir, las que tienen la pendiente situada entre unos valores predeterminados, en general, bastante elevados.

Dentro de todas las líneas obtenidas con pendiente apropiada se eligen las que puedan ser más representativas.

Esas líneas serán las que se utilicen para crear un filtro cuya finalidad sea enderezar esas mismas líneas y las de pendientes similares.

Por otro lado, la función cvHoughLines2(), al igual que las dos anteriores, únicamente analiza una imagen. Por lo tanto se deberá ejecutar independientemente para las dos imágenes, la izquierda y la derecha.

Los resultados proporcionados por el programa tras la aplicación de la función anterior y la selección de las líneas de interés, son las imágenes que se muestran a continuación.

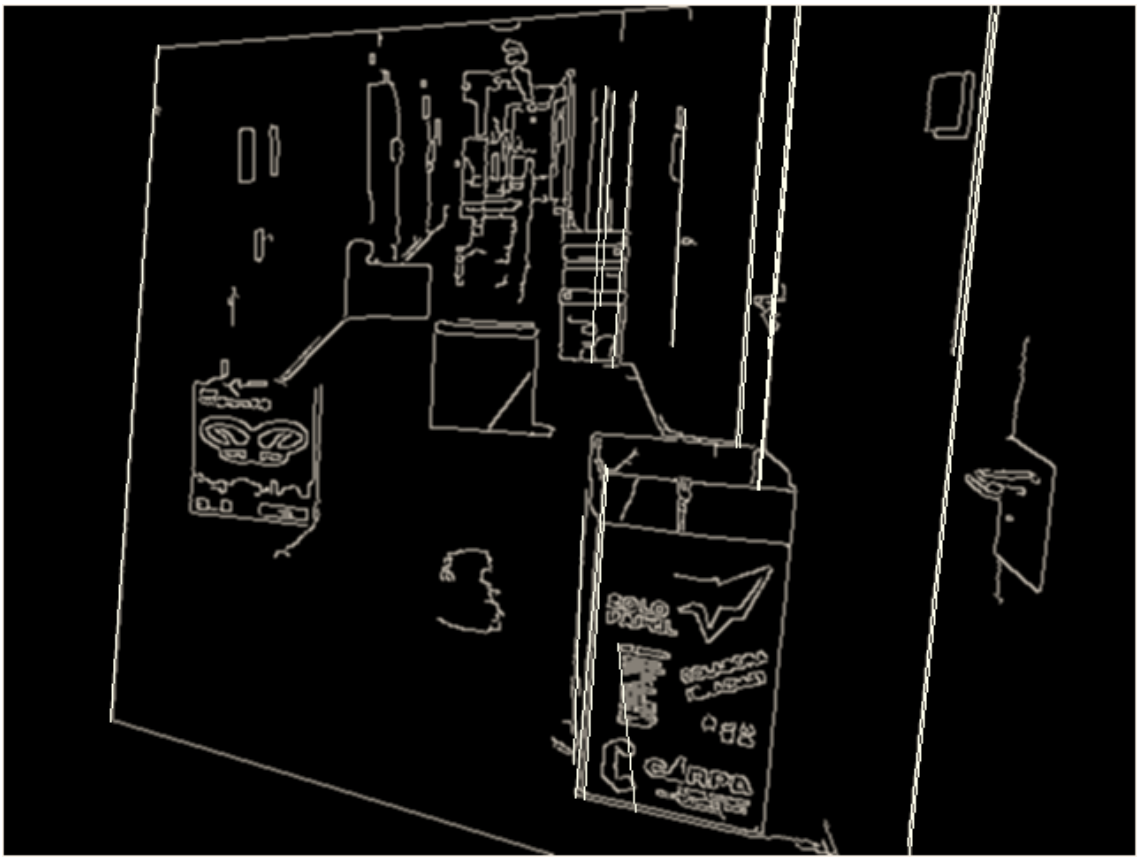


Figura 3.19. *Selección de líneas para enderezar la imagen izquierda*

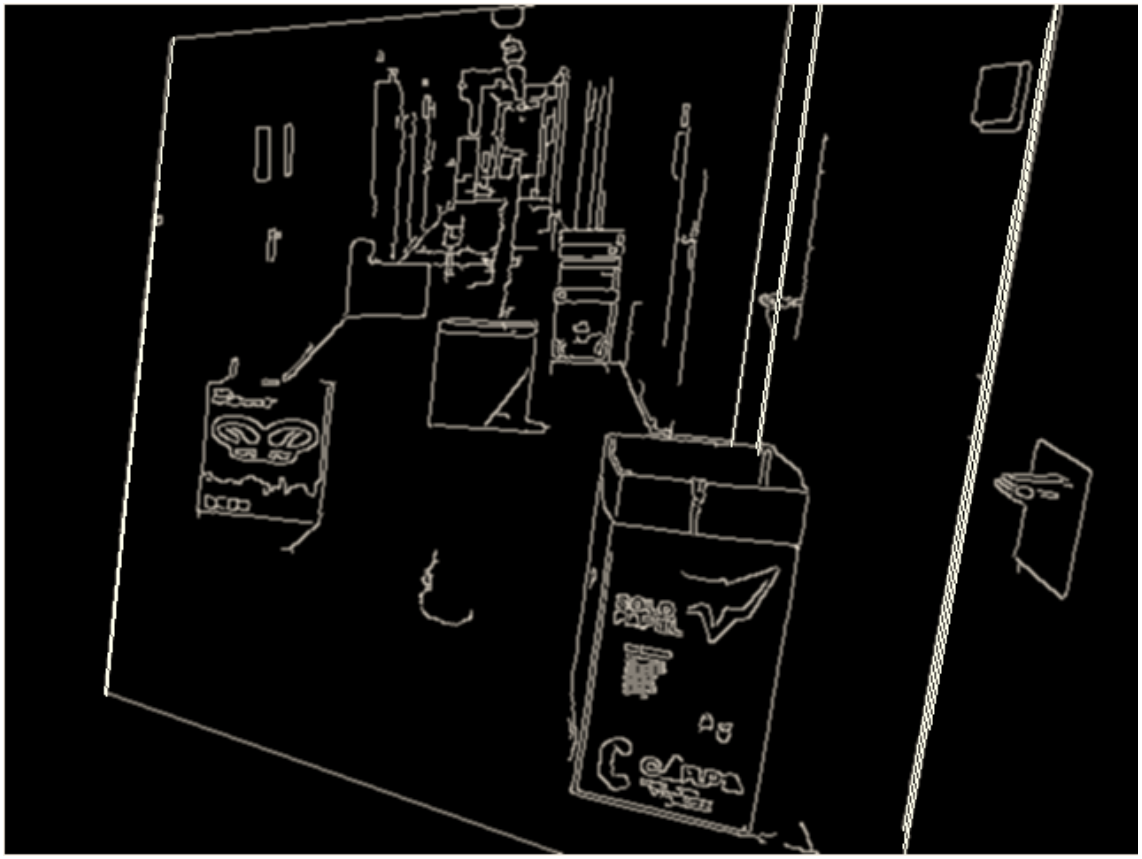


Figura 3.20. *Selección de líneas para enderezar la imagen derecha*

En las figuras 3.19 y 3.29 se pueden observar las líneas escogidas para calcular el filtro. Estas líneas son las representadas con un color amarillo un poco más intenso que el resto.

Una vez se dispone de esta selección de las líneas que deben ser verticales, líneas marcadas en las imágenes anteriores, se procede a calcular la pendiente de las mismas.

La finalidad de obtener la pendiente de las líneas de interés es poder crear el filtro, que, al aplicarlo a las imágenes rectificadas, se tenga como resultado las imágenes sin inclinación.

Para aplicar el filtro a la imagen se ejecuta la función cvWarpAffine(), la cual transforma una imagen fuente según una matriz específica. Esta matriz es el filtro que se desea considerar.

La función cvWarpAffine() dentro del programa junto con los parámetros que la componen, quedarían:

cvWarpAffine (rectificadal, corregidainclinacionl, &_shear);

rectificadal: entrada de la imagen fuente sobre la que se quiere asignar la transformación

corregidainclinacionl: salida con la imagen de destino a la que se le ha aplicado la transformación

&_shear: matriz de transformación

Esta función, al igual que varias de las anteriores, tiene efecto únicamente sobre una imagen. Por lo tanto, habrá que ejecutarla independientemente para cada imagen, la izquierda y la derecha.

Los resultados de aplicar la matriz de transformación a las imágenes rectificadas se pueden ver en las siguientes figuras.

Figura 3.21. *Imagen izquierda sin inclinación*

Figura 3.22. *Imagen derecha sin inclinación*

En la figura 3.23 se muestran juntas las imágenes sin inclinación, la izquierda y la derecha.

Figura 3.23. *Imágenes sin inclinación juntas*

En la figura 3.24 está representada la búsqueda de correspondencias entre las dos imágenes con la inclinación corregida.

Figura 3.24. *Correspondencia de puntos entre las dos imágenes sin inclinación*

De esta forma se puede comprobar que las líneas que han de ser verticales se mantienen como deben estar, proceso que se ha logrado con la ubicación de las últimas funciones.

Y que además no se han perdido las propiedades de las imágenes rectificadas, es decir, un mismo punto de ambas imágenes se encuentra situado en la misma línea horizontal.

Para concluir las comparativas, en la página siguiente, se muestran las imágenes rectificadas (izquierda) junto a las que carecen de inclinación (derecha), separadas por cada cámara, las de la cámara izquierda en primer lugar y las de la derecha justo después.

Una vez que logramos anular la inclinación de las imágenes, ya se está en disposición de calcular correctamente el mapa de disparidad, proceso que se explicará en el próximo apartado.

Figura 3.25. *Imágenes izquierdas rectificada y sin inclinación*

Figura 3.26. *Imágenes derechas rectificada y sin inclinación*

3.6. Mapa de disparidad:

Uno de los últimos pasos del protocolo consiste en hallar un mapa de disparidad del conjunto de imágenes tratadas.

El cómputo del mapa de disparidad, como ya se explicó en el correspondiente apartado teórico, es un paso esencial para el cálculo de las distancias a las que se encuentran los objetos de una imagen con respecto a la cámara con la que fué tomada la fotografía.

La obtención de dicho mapa de disparidad es sencilla, con las librerías OpenCv, una vez se conocen todos los parámetros necesarios.

En el presente proyecto, se va a utilizar la función cvFindStereoCorrespondenceGC() para determinar el mapa de disparidad de la imagen estudiada.

Dicha función requiere dos entradas con las imágenes de interés rectificadas y otra entrada con la información detallada de la manera de proceder para la correcta ejecución de la función. Por otro lado, proporciona una salida con el propio mapa de disparidad.

La función cvFindStereoCorrespondenceGC() dentro del programa junto con los parámetros que la componen, quedarían así:

cvFindStereoCorrespondenceGC (corregidainclinacionl, corregidainclinacionr, dispmapl, dispmapr, GCState);

corregidainclinacionl: entrada con la imagen izquierda ya rectificada y sin inclinación

corregidainclinacionr: entrada con la imagen derecha ya rectificada y sin inclinación

dismapl: salida opcional con el mapa de disparidad izquierdo. Esta imagen tendrá las mismas dimensiones que las imágenes de entrada

dispmapr: salida opcional con el mapa de disparidad derecho. Esta imagen tendrá las mismas dimensiones que las imágenes de entrada

GCState: Estructura de la correspondencia estéreo

Se puede observar que realmente la función devuelve como salida dos mapas de disparidad, aunque a efectos prácticos, en nuestra aplicación solo necesitamos uno de los dos.

Por ello, para el futuro cálculo de la profundidad de los objetos el programa se centrará en el mapa de disparidad derecho. Esta decisión se basa en que dicho mapa queda mejor definido que el izquierdo.

Por otro lado, dentro de la estructura GCState, cabe destacar el campo número de disparidades. Esta variable indica el número de píxeles en que se tienen que buscar correspondencias.

A continuación, en la figura 3.27, se muestra el mapa de disparidad derecho proporcionado por la función. Esta imagen se caracteriza por tener un número de disparidades intermedio, 64.

Figura 3.27. *Mapa de disparidad (64 disparidades)*

Se puede comparar la figura anterior con la que se representa en la siguiente página, en la cual se ve el mapa de disparidad calculado con un número de disparidades excesivamente pequeño, 8, en relación con el de la figura anterior, número de disparidades intermedio, 64.

Figura 3.28. *Mapa de disparidad (8 disparidades)*

Se puede observar la dificultad de obtener resultados concluyentes con un mapa de disparidad poco representativo ó con pocos valores de disparidad, como el indicado en la figura anterior.

4. Conclusiones y Trabajos Futuros

En el presente capítulo se van a exponer las principales conclusiones que se han ido obteniendo a lo largo de la realización del proyecto, tras efectuar los ensayos con el modelo.

Además, se mostrarán algunas líneas de mejora a llevar a cabo, que podrían ser interesantes de estudiar en futuros trabajos, para complementar el algoritmo o hacer más intuitivo su uso.

4.1. Conclusiones:

Durante el desarrollo y ejecución del protocolo se han constatado varios puntos determinantes, los cuales se citan a continuación por orden de aplicación en el proyecto.

Los primeros resultados que se deben analizar son los parámetros de calibración de las cámaras. Estos parámetros se consiguen fácilmente gracias a la implementación de las librerías de OpenCV y a la toma del adecuado número de patrones, variando la orientación de los mismos.

El otro punto relevante en el protocolo es la correcta rectificación de las imágenes. Para, posteriormente, poder hallar un buen mapa de disparidades es imprescindible que las imágenes queden bien rectificadas, es decir, los pares de puntos han de estar horizontales y sin inclinaciones que no estuvieran en las imágenes originales. En caso de que el mapa de disparidad no se obtenga, muy probablemente sea porque no se ha prestado la debida atención a la fase de rectificación.

Por otro lado, tras la ejecución del código se puede afirmar que el tiempo de respuesta del protocolo es bastante lento. Tarda mucho en calcular el mapa de disparidad ante la entrada de un par de imágenes. Esto hace tedioso el trabajo y puede perjudicar a la hora de efectuar pruebas en el algoritmo.

Por último, es importante mencionar que mantener la distancia entre ambas cámaras invariable ayudará en futuras fases de mejora, dado que si hubiera movimiento relativo entre ellas el mapa de profundidades sería muy complejo de calcular. Ésto se consigue en el proyecto con la utilización de la cámara Bumblebee xb3, que lleva integrada de serie las dos cámaras.

4.2. Trabajos Futuros:

En este apartado se van a dejar reflejadas algunas mejoras que han ido surgiendo durante el desarrollo de la aplicación. La finalidad es proporcionar ideas para futuros proyectos o para enriquecer el presente.

La primera modificación sería una optimización del código fuente. Con esta acción se pretenderían reducir los tiempos de repuesta del programa y facilitar la compresión a los usuarios.

Siguiendo la línea de la propuesta anterior, otra posible mejora consistiría en la adecuación de las imágenes rectificadas al espacio total de la imagen para lograr una presentación visual más atractiva. Es decir, al rectificar las imágenes, y tras todas las operaciones posteriores, se ven reflejadas partes negras en un lateral de las imágenes. Dichas partes no contiene ninguna información relevante a la hora de analizar las propias imágenes, por lo que se podrían eliminar.

Otra posible actuación no materializada en el proyecto constaría de ampliar el procedimiento para incluir el análisis de los mapas de profundidad. Este paso proporcionaría la distancia exacta a la que se encuentran los objetos en relación a la cámara, lo cual no solo se utilizaría para completar el protocolo, sino también abriría nuevas puertas para llevar a cabo otras aplicaciones.

5. Costes del proyecto

En el capítulo actual se va a plantear una estimación de los recursos previos necesarios para el diseño y desarrollo del protocolo, objeto principal del presente proyecto.

En primer lugar, se muestran las operaciones llevadas a cabo. Dichas operaciones están desglosadas según la secuencia temporal en la que se deberían realizar.

En segundo y último lugar, se va a proceder a la valoración del propio presupuesto.

5.1. Planificación:

Los puntos generales tenidos en cuenta para la consecución del proyecto son los siguientes:

- Estudio del problema
- Documentación
- Diseño
- Implementación
- Pruebas
- Redacción de la memoria
- Elaboración del presupuesto

Estos puntos pueden desplegarse en subcategorías para presentar mayor detalle.

Por otro lado, dado que todas las tareas expuestas no son materializadas por un solo perfil de trabajador, se ha indicado una distinción entre los dos perfiles de mayor importancia. Es decir, se han separado las tareas según las haya ejecutado un ingeniero analista o un ingeniero programador.

A continuación, en la tabla 5.1, está represento el desglose con la estimación de tiempos, medidos en horas.

Tarea	Tiempo del Ingeniero Analista	Tiempo del Ingeniero Programador
Estudio del problema	**20**	-
Análisis de los requisitos	10	-
Análisis de tecnologías	10	-
Documentación	**60**	-
Diseño	**10**	**4**
Implementación	-	**154**
Instalación del software	-	4
Aprendizaje del software	-	50
Código	-	100
Pruebas y revisiones	-	**50**
Redacción de la memoria	**80**	**30**
Redacción de la memoria teórica	60	-
Redacción de la memoria práctica	20	30
Elaboración del presupuesto	**2**	-
Total	**172**	**238**

Tabla 5.1. *Desglose de tareas del proyecto*

Todos los tiempos que se pueden observar en la tabla 5.1 son estimaciones aproximadas.

5.2. Presupuesto:

El coste de recursos humanos, según las valoraciones definidas en el apartado anterior y aplicando salarios medios (facilitados por la web [31]), quedaría de la siguiente manera:

Recursos Humanos	Importe/ Hora	Nº Horas	Importe Total
Ingeniero Analista	36€/h	172	6.192€
Ingeniero Programador	21€/h	238	4.998€
Total			**11.190€**

Tabla 5.2. *Coste de recursos humanos del proyecto*

Una ventaja del proyecto, en relación a la cuantía económica a desembolsar, es que se puede desarrollar sin necesidad de pagar por el software que se utilice. Ésto es debido a que el software requerido es de acceso libre.

En cuanto a los materiales y al equipo hardware sí habría que disponer de ellos. Para el caso de que hubiera que comprarlos todos, las estimaciones de estos gastos se observan en la tabla 5.3.

Materiales	Importe Total
Ordenador portátil (Dell Inspiron + ratón + soporte)	600€
Cámara (Bumblebee xb3)	3.000€
Trípode (Giottos MT9242)	80€
Otros materiales (cables, gastos de oficina…)	100€
Total	**3.780€**

Tabla 5.3. *Costes de materiales del proyecto*

De acuerdo con los resultados de las operaciones anteriores, se puede afirmar que el proyecto saldría por el precio aproximado que aparece reflejado en la siguiente tabla, 14.970€.

Presupuesto	Importe Total
Coste de recursos humanos	11.190€
Coste de materiales	3.780€
Total	**14.970€**

Tabla 5.4. *Coste total del proyecto*

6. Bibliografía

[1] Aracena Pizarro, Diego; Campos, Pedro; Tozzi, Clésio Luis. "Comparación de Técnicas de calibración de Cámaras Digitales". *Revista Facultad de Ingeniería – Universidad Tarapacá*, vol. 13, nº 1 (2005), p. 57-67

[2] Ballard, Dana H.; Brown, Christopher M. *Computer Vision*. Englewood Cliffs, New Jersey: Prentice Hall, 1982

[3] Bradski, Gary; Kaehler, Adrian. *Learning OpenCV*. United States of America: O'Reilly, 2008

[4] Burger, Wilhelm; Burge, Mark J. *Principles of Digital Image Processing: Core Algorithms*. London: Springer, 2009

[5] Cyganek, Boguslaw; Siebert, J. Paul. *An Introduction to 3D Computer Vision Techniques and Algorithms*. Chichester, England: Wiley, 2009

[6] Davies, E. R. *Machine Vision: Theory, algorithms, Practicalities*. Tercera edición. Amsterdam: Morgan Kaufmann (Elsevier), 2005

[7] Escalera, Arturo de la; Armingol, José María; Pech, José Luis; Gómez, José Julián. "Detección Automática de un Patrón para la calibración de Cámaras". *Revista Iberoamericana de Automática e Informática Industrial*, vol. 7, nº 4 (2010), p. 83-94

[8] Escalera, Arturo de la. *Visión por Computador: Fundamentos y Métodos*. Madrid: Prentice Hall, 2006

[9] Evans, Christopher. *Notes on the OpenSURF Library*. 2009

[10] Faugeras, O. *Three dimensional computer vision: a geometric viewpoint*. Cambridge: MIT Press, 1993

[11] Forsyth, David A.; Ponce, Jean. *Computer Vision: A Modern Approached*. Segunda edición. Pearson Education, 2011

[12] Ghali, Sherif. *Introduccion to geometric computing*. London: Springer, 2008

[13] González Jiménez, Javier. *Visión por Computador*. Madrid: Paraninfo, 2000

[14] Hartley, R.; Zisserman, A. *Multiple view geometry in computer vision*. Segunda edición. Cambridge: Cambridge University Press, 2004

[15] Igual, Raúl; Medrano, Carlos. *Tutorial de OpenCV*. Teruel: Computer Vision Lab, 2008

[16] Laganière, Robert. *OpenCV 2 Computer Vision Application Programming Cookbook*. Packt Publishing, 2011

[17] O'Gorman, Lawrence; Sammon, Michael J.; Seul, Michael. *Practical Algorithms for Image Analysis: description, examples, programs, and projects*. Segunda edición. Cambridge: Cambridge University Press, 2008

[18] Pajares Martinsanz, Gonzalo; Cruz García, Jesús M. de la. *Ejercicios resueltos de Visión por Computador*. Madrid: Ra-Ma, 2007

[19] Pajares Martinsanz, Gonzalo; Cruz García, Jesús M. de la. *Visión por Computador: Imágenes digitales y aplicaciones*. Segunda edición. Madrid: Ra-Ma, 2007

[20] Parker, J. R. A*lgorithms for Image Processing and Computer vision*. New York: Wiley Computer Publishing, 1997

[21] Pitas, I. *Digital Image Processing Algorithms and applications*. New York: Wiley Computer Publishing, 2000

[22] Ritter, Gerhard X.; Wilson, Joseph N. *Handbook of Computer Vision Algorithms in Image Algebra*. Boca Raton: CRC Press, 1996

[23] Riveros Guevara, Adriana; Salas López, Cindy Natalia, Solaque Guzmán, Leonardo. "Aproximación a la Navegación Autónoma de una Plataforma Móvil, mediante Visión Estereoscópica Artificial". *Ciencia e Ingeniería neogranadina*, vol.22, nº 2 (2012), p. 111-129

[24] Seul, Michael; O'Gorman, Lawrence; Sammon, Michael J. *Practical Algorithms for Image Analysis: description, examples, and code*. Cambridge: Cambridge University Press, 2001

[25] Sonka, Milan; Hlavac, Vaclav; Boyle, Roger. *Image processing, analysis, and machine vision*. Tercera edición. Toronto: CENGAGE Learning, 2008

[26] Vernon, David. *Machine Vision: automated visual inspection and robot vision*. New York: Prentice Hall, 1991

6.1. Webgrafía:

[27] *Amazon* [en línea]. Precio de los materiales del proyecto. Disponible en: http://www.amazon.com

[28] Bouguet, Jean-Yves. *Camera Calibration Toolbox for Matlab* [en línea]. 2 de Diciembre de 2013. Disponible en: http://www.vision.caltech.edu/bouguetj/calib_doc/

[29] *OpenCV wiki* [en línea]. Sitio web de OpenCV. Disponible en: http://code.opencv.org

[30] *Point grey* [en línea]. Precio de los materiales del proyecto. Disponible en: http://www.ptgrey.com

[31] *tufunción* [en línea]. Precio de la mano de obra del proyecto. Disponible en: http://www.tufuncion.com/trabajo-programador

[32] Willow Garage [en línea]. Sitio web de OpenCV. Disponible en: http://www.willowgarage.com/pages/software/opencv

Anexo A: Cámara Bumblebee xb3

Algunas características técnicas de interés acerca de la cámara Bumblebee xb3, prestada por el Departamento de Automática de la Universidad Carlos III de Madrid para la realización del proyecto, se muestran en la siguiente tabla.

Especificación	
Línea Base	12cm/ 24cm
Dimensiones	277 x 37 x 41,8 mm
Peso	505g
Distancia focal f_x	1,2656 x resolución en columnas
Distancia focal f_y	1,6874 x resolución en filas
Resolución en Columnas	640 píxeles
Resolución en Filas	480 píxeles

Tabla A.1. *Características de la cámara Bumblebee xb3*

Figura A.1. *Cámara Bumblebee xb3*